DISS. ETH NO. 16740

Locality, Scheduling, and Selfishness:

Algorithmic Foundations of Highly Decentralized Networks

A dissertation submitted to the

SWISS FEDERAL INSTITUTE OF TECHNOLOGY ZURICH

for the degree of

Doctor of Sciences

presented by

THOMAS MOSCIBRODA

Dipl. Inf.-Ing., ETH Zürich

born 11.09.1979

citizen of

Malters LU

accepted on the recommendation of

Prof. Dr. Roger Wattenhofer, examiner
Prof. Dr. Christos H. Papadimitriou, co-examiner
Prof. Dr. David Peleg, co-examiner

2006

Abstract

Large-scale and highly decentralized networks such as the Internet have emerged as arguably the most complex computer systems. The size of such networks, as well as their dynamic, socio-economic, and often wireless nature brings about a large variety of challenging problems. Achieving efficient and provably robust algorithmic solutions for these problems, in turn, necessitates the development and employment of novel techniques and methods. This dissertation studies three major concepts that stand out as being of particular importance and interest in this context: *locality*, *wireless communication*, and *selfishness*.

In large-scale networks, gathering information about the whole network topology is either too resource consuming or simply impossible due to mobility, dynamics, or churn. Hence, no node is typically able to collect or maintain a global state about the network and each node has to base its decision on local information only. This discrepancy between the need to achieve global efficiency and the limitation to local knowledge motivates the study of *local algorithms* and *local computation*. The first part of this dissertation investigates the possibilities and limitations of local computation in the classic message passing model of distributed computing. We present near-tight upper and lower bounds on the achievable trade-off between the amount of local knowledge and the resulting global solution for a variety of fundamental network problems.

The integration of *wireless devices* into the Internet, and the advent of wireless multi-hop networks such as *ad hoc and sensor networks* poses numerous algorithmic challenges. This dissertation investigates the distributed complexity of computing network coordination structures such as clusterings and colorings in models that closely capture unstructured wireless multi-hop networks. We then define and study the *scheduling complexity of wireless networks* in a physical model of wireless communication. This measure describes the theoretically achievable performance of any scheduling protocol and allows to characterize and analytically evaluate existing protocols from a worst case perspective.

Hosts in the Internet or peers in a peer-to-peer network are typically governed by socio-economic agents whose main interest is not the benevolent optimization of the network's entirety, but rather the maximization of their own benefit. In other words, participating agents may be selfish, rather than acting in a coordinated manner that optimizes social welfare. The final part of this thesis analyzes the impact of selfish and potentially malicious behavior on the efficiency of peer-to-peer and other decentralized computer networks.

Zusammenfassung

Grossflächige, dezentrale Netzwerke wie das Internet haben sich zu äusserst komplexen Computersystemen entwickelt. Diese Netzwerke sind typischerweise dynamisch, beinhalten drahtlose Verbindungen und werden oft von unabhängigen, eigennützig agierenden Netzwerk-Knoten betrieben. Um die daraus resultierenden Probleme zu lösen, bedarf es algorithmischer Verfahren die beweisbar effizient sind und auch in worst-case Szenarien akzeptable Resultate liefern. Diese Dissertation befasst sich mit drei zentralen Problemfeldern von grossen, dezentralen Netzwerken: *Lokalität*, *drahtlose Kommunikation* und das *eigennützige Verhalten* einzelner Netzwerkteilnehmer.

In grossen und dynamischen Netzwerken ist es für einzelne Knoten nicht möglich, den Zustand des gesamten Netzwerkes zu kennen. Typischerweise ist kein Knoten in der Lage, globale Informationen über das Netzwerk einzusammeln oder zu verwalten. Trotz dieser Beschränkung auf *lokales Wissen* müssen die Knoten jedoch die Leistungsfähigkeit des gesamten Netzwerkes garantieren oder globale Ziele wie Stabilität und Effizienz erreichen. Das verteilte Berechnen und Optimieren globaler Eigenschaften basierend auf lokaler Information ist demnach eine der zentralen Fragestellungen im Bereich der verteilten Algorithmen. Der erste Teil dieser Dissertation analysiert die fundamentalen Möglichkeiten solcher *lokalen Algorithmen* im klassischen Message-Passing Model des verteilten Rechnens. Wir charakterisieren den theoretisch erreichbaren Trade-off zwischen the Menge an lokalem Wissen und der resultierenden globalen Lösung für wichtige Netzwerk-Probleme mittels nahezu übereinstimmenden oberen und unteren Schranken.

Die Einbindung von drahtlosen Geräten ins Internet und das Entstehen von Ad-Hoc-, Sensor- und anderen drahtlosen Multi-Hop-Netzwerken wirft ebenfalls eine Vielzahl algorithmischer Fragestellung auf. Der zweite Teil dieser Dissertation untersucht die Komplexität des verteilten Berechnens wichtiger Netzwerk-Strukturen wie beispielsweise Färbungen und "Clusterings" in Modellen, welche die physikalische Realität drahtloser Netzwerke möglichst genau abbilden. Des Weiteren wird die "Scheduling Complexity of Wireless Networks" im physikalischen Modell drahtloser Kommunikation definiert. Dieses Mass beschreibt die maximal erreichbare Effizienz von Scheduling Protokollen und erlaubt ausserdem eine analytische Charakterisierung des worst-case Verhaltens existierender Protokolle.

Computer im Internet oder Peers in einem Peer-to-Peer Netzwerk werden typischerweise von unabhängigen, sich ökonomisch verhaltenden Individuen betrieben und kontrolliert. Das Hauptinteresse dieser Netzwerkteilnehmer ist dabei oft nicht die Leistungsoptimierung des gesamten Netzwerkes, sondern die Maximierung des eigenen Profits. Mit anderen Worten, dezentrale Netzwerke werden oft von Teilnehmern gebildet, welche sich eigennützig verhalten, ohne Berücksichtigung des Gesamtwohles im Netzwerk. Der letzte Teil dieser Dissertation analysiert den Einfluss eigennütziger und möglicherweise sogar bösartiger Teilnehmer auf die Leistungsfähigkeit und Effizienz von Peer-to-Peer- und anderen dezentralen Computer Netzwerken.

Acknowledgements

I would like to express my gratefulness to my advisor Roger Wattenhofer for guiding me through my thesis and for encouraging me to pursue a Ph.D. in the Distributed Computing Group in the first place. During my time as your student, you have not only invested a lot of time and effort in supporting my studies, but you have also made me acquainted with the customs of academic life and research. It has been a tremendous pleasure to work with you and I am proud to be your scientific offspring!

I would also like to express my profoundest gratitude to my co-examiners Christos H. Papadimitriou and David Peleg for their willingness to read through the thesis and serve on my committee board. It has been a great pleasure and honor to receive positive comments from two such renowned researchers.

Many thanks go to all members of the Distributed Computing Group for creating a warm and inspiring atmosphere during the last two and a half years. First and foremost, I would like to thank my office mate Pascal von Rickenbach for being my faithful and good companion during many years of undergraduate studies, the master thesis, and finally the time as a Ph.D. student. I have always enjoyed your delightful sense of humor and your positive and candid way of reacting to the ups and downs of life. I also thank you for our countless discussions about everything and nothing that have been a source of inspiration and an enrichment of my daily work hours. Also, I would like to thank Aaron Zollinger—the master of poetry and cultivated language—for being a true role model in so many ways, not only when it comes to table-soccer; Fabian Kuhn for successfully guiding and supervising me through my Master thesis and for our very interesting, intensive, and prolific research collaboration ever since; Keno Albrecht for his amazing patience in solving my often self-made computer problems, his dynamic style of playing table soccer, and for his spam-filter "Spamato" that has helped me cope with spam; Regina O'Dell for teaching me about the marvels of birth and breast-feeding and, generally, for her wit and stamina in defending women's perspectives in our male-dominated group; Nicolas Burri for his great sense of humor and for conveying some of the subtleties of modern didactics to me, and for sharing my passion for table soccer. Stefan Schmid for a productive research collaboration, for successfully challenging my blind-chess skills, and for his mastery in creating scientific figures.

Furthermore, I would like to thank all the newer members of the DCG group—Roland Fluri, Michael Kuhn, Yves Weber, Thomas Locher, Olga Goussevskaia, and Yvonne Anne Oswald—for providing a warm and inspiring research and work environment during the final stages of my PhD. It is good to know that the future of the DCG group is in the hands of so many great colleagues.

Above all, I would like to express my gratitude to my family. I thank my brothers, Stefan and Kentaro, for interesting and adventurous discussions about life and the world's future. I also thank my parents-in-law Takako and Izumi for their warmth and hospitality during my stays in Japan, where many of my research results originated and developed. I am forever grateful

and indebted to my parents Rosa and Josef for their love and care. From the very beginning, you have provided me with support and guidance on my way that has now led me to where I stand today.

Finally and most importantly, my deepest and most sincere thanks belong to my beloved wife Hiroko. Your incessant love and encouragement has been my motivation to work hard throughout the thesis and before. I am—and will always be—grateful to you for standing by me through all ups and downs, relaxing and (unfortunately too often) stressful times, at home and on one of our wonderful trips allover the world. After all, I know that it is *you* who brings out the best in me.

Contents

Chapter 1

Introduction

In recent years, large-scale and highly decentralized networks such as the Internet, wireless ad hoc and sensor networks, or peer-to-peer networks have emerged as the most interesting and challenging computer systems. The sheer size of such networks as well as their inherently dynamic nature brings about a large number of problems that require efficient and robust algorithmic solutions, which, in turn, can only be obtained by developing and employing a variety of novel methods and techniques.

One new dimension added to the focus of computer scientists is the socio-economic nature of modern networks. Hosts in the Internet or peers in a peer-to-peer network may have no incentive to altruistically follow a pre-defined protocol. Instead, such entities may prefer to selfishly deviate from protocols in order to increase their own benefit or reduce their own cost. But even in systems consisting entirely of benevolent participants, numerous challenges remain. In particular, one of the key distinctions of large-scale computer networks is the absence of global knowledge, leaving *local computation* as the only acceptable method for performing global tasks or maintaining global structures and equilibria. In dynamic and mobile settings, for instance, nodes may be unable to obtain a large set of up-to-date information about the state of the network. Nonetheless, algorithms are expected to adapt to topology and state changes in a quick and light-weight manner, even at the cost of a potentially sub-optimal solution. Also, in view of the ever-growing complexity of large-scale computer networks, *managing* these systems has become an increasingly crucial part of these network's success. One way to enhance the manageability (and at the same time reduce the vulnerability) of large-scale computer networks is to construct them in a *self-organizing*, self-healing, and potentially self-protecting way, which, again, asks for *local distributed algorithms* that are capable of quickly reacting to changes.

Two concepts thus stand out as being of particular algorithmic importance in large-scale and highly decentralized networks: *locality* and *selfishness*. And as it turns out, studying either of these concepts requires novel algorithmic techniques and leads to fascinating insights into the nature and complexity of modern computer networks.

In large computer networks (as well as other networks such as the human brain or society), no participating entity has the ability to achieve *global knowledge* about the entire state or topology of the network. Instead, each node can maintain only a restricted amount of *local information* about its neighborhood. The importance of an algorithmic theory of *locality* and *local computation* therefore stems from the discrepancy between local knowledge and global objective: in spite of the restriction to local information only, the entirety of the network is supposed to achieve a global goal, to keep a global equilibrium, or (at least) to maintain a reasonable global performance. Part I of this thesis aims at studying the fundamental consequences implied by the restriction to local knowledge. Specifically, this part is geared towards shedding light into the local computability or approximability of important network coordination tasks. That is, it explores the trade-off between the amount of local knowledge (or alternatively, between the time required to gather such knowledge) and the quality of the resulting global solution.

Part IV of this thesis studies the other important concept of Internet-like networks: *selfishness*. As motivated above, the participating hosts in the Internet or a peer-to-peer network may not necessarily be considered as altruistic in the sense that they cooperate to achieve a common goal. Instead, each network node may be primarily interested in optimizing its own utility. In order to characterize and analytically capture the implications of selfish and rational behavior in distributed systems, researchers have introduced micro economic models in computer science. Part IV applies these techniques two specific problem scenarios found in peer-to-peer networks and the Internet.

Another aspect of modern computer networks is the integration of *wireless devices*—and hence, wireless communication links—into the Internet. Technologies such as WLAN or UMTS have rendered ubiquitous Internet access a reality in many areas of public life. A crucial aspect of communication in a wireless medium is *scheduling*. If too many devices in physical proximity transmit simultaneously, the interference caused by these transmissions will prevent an intended receiver from receiving the signal, i.e., the message is lost. On the other hand, if too few nodes transmit at the same time, valuable bandwidth is wasted and the overall throughput suffers. Hence, the dilemma faced by any scheduling protocol is that neither selecting too many nor too few devices for concurrent transmission is acceptable. Part III of this thesis focuses on the theoretical possibilities and limitations of scheduling and MAC layer protocols in wireless networks. Based on a physical model of signal propagation, it is shown that the throughput achieved by intuitive and often employed MAC layer and scheduling protocols can be drastically sub-optimal even for simple communication tasks. This is the case even for protocols that can be shown to be optimal under frequently studied, yet idealistic graph-based models of communication.

Mobile phone technology or Wireless LAN still require the availability of a statically installed backbone infrastructure. Typically, it is only the last hop—from the access point to the end device—that uses a wireless link. An even more intrepid idea, however, is the construction of self-organizing *multi-hop wireless networks* that are capable of working in the *absence of any built-in infrastructure*. Such wireless *ad hoc and sensor networks* have

been envisioned in a large number of application fields. The prospect of aggregating sensor nodes into sophisticated computation and communication infrastructures is bound to have a significant impact on a wide array of scientific, social, and industrial applications. Moreover, there are a growing number of real (even commercial) systems that are being built, ranging from monitoring and surveillance, to medical applications, the observation of biological and chemical processes, disaster relief, and the construction of community-based mesh networks.

These networks are formed by autonomous nodes that communicate via radio, without any additional a-priori infrastructure. Typically, if two nodes are not within mutual transmission range, they communicate through intermediate nodes relaying their messages. In other words, the nodes themselves provide and maintain a virtual communication infrastructure in a distributed manner. From an algorithmic point of view, dealing with these networks offers a multitude of interesting and challenging problems. One of the key peculiarities of these networks is that they are supposed to work under harsh environments, especially during their deployment phase. Achieving efficient algorithmic solutions to the outstanding problems of initializing and structuring ad hoc and sensor networks therefore requires a subtle modeling of the specific characteristics encountered in these networks. Part II of this thesis explores the impact of harsh realities faced by wireless ad hoc and sensor networks on the distributed complexity of solving network coordination problems. Based on a novel *unstructured radio network model*, we present provably efficient protocols for setting up network coordination structures such as clusterings and colorings in ad hoc and sensor networks, thus—in a sense—providing an "unstructured wireless counterpart" to many of our results obtained in Part I of this thesis.

Chapter 2

Definitions and Preliminaries

This chapter defines terms that are going to be used throughout this work. Notation that is peculiar to a specific chapter will be introduced in the corresponding model sections.

We begin with some terminology regarding graphs. When modeling a network as a graph $G = (V, E)$, the vertices $V = \{v_1, v_2, \ldots, v_n\}$ correspond to nodes in the network and there is a bidirectional communication link between two nodes v_i and v_j, if $(v_i, v_j) \in E$. As customary, the number of nodes in the graph is denoted by $n = |V|$. The distance measured in hops (called the *hop-distance*) between two nodes v_i and v_j is denoted by $d_G(v_i, v_j)$. The graph's *diameter* D is the maximal hop-distance between any two nodes in the graph, i.e., $D = \max_{u,v \in V} d_G(u, v)$.

The notion of neighborhoods plays an important role in large parts of this thesis. The *neighborhood* of a node $v \in V$, denoted by $\Gamma(v)$, is the set of v's neighbors in the graph. Unless otherwise stated, this definition is extended to include also v itself in $\Gamma(v)$. Formally, the neighborhood of node v is

$$\Gamma(v) \ := \ \{v\} \cup \{w \mid (v, w) \in E\}.$$

When dealing with local distributed algorithms, we will make use of neighborhoods of a particular depth. For some constant $\rho \geq 0$, the *ρ-neighborhood* $\Gamma_\rho(v)$ of a node $v \in V$ is defined as the union of nodes that are at most ρ hops away from v in G:

$$\Gamma_\rho(v, G) \ := \ \{v\} \cup \{w \mid d_G(v, w) \leq \rho\}.$$

Whenever G is clear from the context, we simply write $\Gamma_\rho(v)$. Finally, note that $\Gamma_1(v) = \Gamma(v)$ and $\Gamma_0(v) = \{v\}$.

The *degree* δ_v of a node v is the number of neighbors of v in G, i.e., $\delta_v := |\Gamma(v)| - 1$. Some of our asymptotic results presented in later chapters will depend not only on the number of nodes n, but also on the *maximal degree* Δ in the graph, which is the maximal number of nodes in the neighborhood of a node $v \in V$. Formally, $\Delta := \max_{v \in V} \delta_v$. Clearly, it holds that $1 \leq \Delta \leq n$.

We call two nodes $u, v \in V$ *independent* if they are not adjacent in the graph G, i.e., $(u, v) \notin E$. An *independent set* $S \subseteq V$ in a graph is a set of nodes that are mutually independent, i.e., no two nodes in S are neighbors.

Sometimes, we will consider metric spaces instead of graphs. A *metric space* M is a pair (X, d), where X is a set of points and $d : X \times X \to [0, \infty)$ is a *distance function* satisfying the properties $\rho(u, v) = \rho(v, u)$ and the triangle inequality, $\rho(u, w) + \rho(w, v) \leq \rho(u, v)$ for all $u, v, w \in X$. A finite metric space can be represented as a complete graph on $|X|$ vertices, with edge lengths between pairs of vertices equal to the distance between the respective points in the metric space. In a metric space $M = (X, d)$, the *ball* of radius r around node v, denoted by $B_r(v)$, consists of all points that are at most at distance r of v, formally $B_r(v) = \{u \in X | d(u, v) \leq r\}$.

If not specified otherwise, $\log x$ corresponds to the logarithm to base 2, whereas the natural logarithm of x is written as $\ln x$. An important function in distributed complexity is the so-called *log-star function* $\log^* x$. Let the i^{th} logarithm of x be the value which results from applying the logarithm to x i times. Formally, for an integer $i > 0$, the i^{th} logarithm of x is defined as

$$\log^{(i)} x \ := \ \begin{cases} \log n & , \quad i = 1 \\ \log\left(\log^{(i-1)} n\right) & , \quad i > 1 \end{cases}$$

The log-star function is then defined as the number of times the logarithm has to be taken before the value decreases below 2.

$$\log^* x \ := \ \min\{i \mid \log^{(i)} n \leq 2\}.$$

Throughout the thesis, we will study combinatorial optimization problems and (distributed) algorithms, which compute solutions to those problems. In order to measure the quality of the achieved solution, it is compared to an optimal solution. Consider a minimization problem. Let ALG_I be the value of the solution produced by an algorithm $\mathcal{A}$ for input I. The value of the optimal solution for I is denoted by OPT_I. The *approximation ratio* α of algorithm $\mathcal{A}$ is defined as

$$\alpha \ := \ \sup_I \frac{ALG_I}{OPT_I}.$$

For maximization problems, the approximation ratio is analogously defined as $\alpha := \max_I \frac{OPT_I}{ALG_I}$. If adapting constant parameters in the algorithm allows to achieve an approximation ratio of $1 + \epsilon$ for every constant $\epsilon > 0$, the algorithm is called an *approximation scheme*.

As far as randomized algorithms and probabilistic arguments are concerned, we use the term "with high probability" for events that occur with probability $1 - n^{-c}$ for a constant c, which can be made arbitrarily large by setting the corresponding parameters to large enough values. Typically, we require $c \geq 1$.

The chapter is concluded with some well-known mathematical theorems and facts. When bounding probabilities, the *Chernoff Bounds* (see for instance [181]) describe the tail behavior of the distribution of the sum of independent Bernoulli experiments.

Theorem 2.1 (Upper Tail). *Let $X_1, X_2, \ldots, X_N$ be independent Bernoulli variables with probability $Pr[X_i = 1] = p_i$. Let $X := \sum_i X_i$ be the sum of the X_i and let $\mu := E[X] = \sum_i p_i$ be the expected value for X. For $\delta > 0$, it holds that*

$$Pr[X > (1+\delta)\mu] \ < \ \left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right)^{\mu}.$$

Theorem 2.2 (Lower Tail). *Let $X_1, X_2, \ldots, X_N$ be independent Bernoulli variables with probability $Pr[X_i = 1] = p_i$. Let $X := \sum_i X_i$ be the sum of the X_i and let $\mu := E[X] = \sum_i p_i$ be the expected value for X. For $\delta \in (0, 1]$, it holds that*

$$Pr[X < (1-\delta)\mu] \ < \ \left(\frac{e^{-\delta}}{(1-\delta)^{(1-\delta)}}\right)^{\mu} \ < \ e^{-\mu\delta^2/2}.$$

The following fact was proven in [138] and relates the product of a set of (small-enough) probabilities to their sum.

Fact 2.1. *Given a set of probabilities $p_1 \ldots p_n$ with $\forall i : p_i \in [0, \frac{1}{2}]$, the following inequalities hold:*

$$\left(\frac{1}{4}\right)^{\sum_{k=1}^{n} p_k} \ \leq \ \prod_{k=1}^{n} (1 - p_k) \ \leq \ \left(\frac{1}{e}\right)^{\sum_{k=1}^{n} p_k}.$$

Finally, the following inequalities are frequently used.

Fact 2.2. *For all n, t, with $n \geq 1$ and $|t| \leq n$, it holds that*

$$e^t \left(1 - \frac{t^2}{n}\right) \ \leq \ \left(1 + \frac{t}{n}\right)^n \ \leq \ e^t.$$

Part I

Local Computation

Chapter 3

Locality in Distributed Computations

Modern large-scale distributed systems such as peer-to-peer networks, wireless ad hoc and sensor networks, or the Internet have in common that they consist of individual nodes which can directly communicate with a limited number of neighboring nodes only. In such networks, no node is typically able to collect global information from the entire network or maintain a global state, because gathering information about the whole network topology is either too resource consuming or simply impossible due to mobility, dynamism, or churn. In spite of these inherent limitations to *local knowledge* and *local information*, networks are expected to compute a good global solution, maintain stability, or achieve a reasonable performance. This discrepancy between the need to achieve global efficiency or stability and the impossibility of basing individual decisions global knowledge motivates the study of *local algorithms* and *local computation*.

Consider nodes in wireless ad hoc and sensor networks, which are typically severely constrained with regard to bandwidth, memory, computing power, and energy. As a consequence, protocols running on these devices should require as little communication as possible and—in view of constantly changing topologies—should run as fast as possible. Both goals can only be achieved by *distributed algorithms* that require only few (ideally a constant number of) rounds of communication until termination. In such a local algorithm, each node must take its decision without global coordination or global knowledge. Likewise, there are numerous practical settings in which employing a local algorithm is the only acceptable choice when designing scalable network protocols that can cope with dynamism and mobility.

What exactly is a *local algorithm*? Informally, a distributed algorithm or a computation can be considered local if each individual node requires only knowledge about its vicinity in the network graph (as opposed to global knowledge) in order to collectively achieve an efficient solution to a global

problem. Technically speaking, we call a distributed algorithm *local* if it is run by individual nodes in the network and the algorithm's running time is (significantly) smaller than the network's diameter. Inevitably, if the number of communication rounds is smaller than the network's diameter, each node can base its decision only on partial, in fact *local knowledge*.

The following fundamental questions arise from these observations: *What can and what cannot be computed locally?* Or more technically, how good can a global solution be if the local decisions at individual nodes are based on restricted local views? What is the price of not having global coordination and knowledge? And, how much does the global solution improve if each node is given information about one additional "hop" of its neighborhood? In other words, what is the *value of local information*?

Besides being of practical importance in networking, a profound understanding of the possibilities and limitations of *local computation* appears to be of theoretical interest in various areas, including distributed computing, graph and approximation theory, and information theory. A theory of local computation may even turn out to be an instrumental ingredient for theoretical studies of social networks or the human brain, because, after all, neurons in the brain or individuals in society can also base their decisions on partial and local knowledge about the entirety of the global state only.

In Part I of this thesis, we examine the impact of locality and local computation on the *complexity of distributed problems* in the classic message passing model of distributed computing. In this model, which we describe in detail in Chapter 4, each node of a graph represents a processor and has a unique identifier (e.g. IP address), two nodes can communicate if they share an edge in the graph. In the standard synchronous model where time is divided into rounds and each node can send a message to all of its neighbors in every round, there is a strong correspondence between the *locality* of local computation on the one hand (*graph theory*), and the *time complexity* of a distributed algorithm on the other hand (*distributed computing*). Specifically, in k rounds of communication, a node can gather information from at most its k-hop neighborhood in the graph.

While *global optimization* with full information has been studied by researchers in great detail for decades, surprisingly little is known about the foundations of *local computation* and many of the major achievements in this field (e.g. by Linial [166], Naor and Stockmeyer [184], Papadimitriou and Yannakakis [193], or Peleg [196]) date back more than 15 years.

While certain combinatorial problems such as the MST or Steiner problems appear to be inherently global, there are others that seem much more local in nature. Consider for example the traditional graph theory problem of computing a *minimum dominating set* (MDS), or even simpler, a *minimum vertex cover* (MVC) in a graph. In a global setting, the complexity of both MDS and MVC is well understood. But what if nodes are processors of a network and only have local knowledge? How much do nodes need to know about the entirety of the network in order to come up with a feasible solution and a good global approximation? Specifically, what is the achievable trade-off between the locality of a distributed computation (number of communication rounds) and the quality of the achieved global

solution (e.g. approximation ratio)? Chapter 5 defines MVC, MDS, facility location, and other related local problems, including exact combinatorial problems such as the maximal independent set (MIS) problem which can trivially be solved by sequential algorithms, but becomes much harder in the distributed case. Subsequent Chapter 6 presents distributed algorithms that achieve non-trivial approximation guarantees for various local problems even for an arbitrary constant running time k, i.e., when the nodes' information about the network is limited to their k-hop neighborhood.

Intuitively MVC appears to be perfectly suited for a local algorithm: A node should be able to decide whether to join the vertex cover by communicating with its neighbors a few times. Very distant nodes seem to be superfluous for its decision. The fact that there is a simple greedy algorithm which approximates MVC within a factor 2 in the *global* setting, additionally raises hope for a fast local algorithm. Surprisingly, however, there is no such local distributed algorithm. In Chapter 7, we prove strong lower bounds on the achievable global solution of MVC and MDS given that each node must base its decision on local information from its k-hop neighborhood only. This, in turn, implies novel time lower bounds for distributed algorithms and, in particular, hardness of distributed approximation lower bounds. Moreover, from these *approximability lower bounds*, we can derive *locality lower bounds* for two important exact problems in distributed computing: *maximal matchings* and *maximal independent sets* (MIS). Since all lower bounds hold even in the cases of unbounded messages and complete synchrony, the lower bounds are a true consequence of *locality* limitations, and not merely side-effects of congestion, asynchrony, or limited message size as frequently encountered in distributed computing. As such, our results stand in line with the only previous unconstrained locality lower bound in the message passing model of distributed computing [166]. Particularly, our time lower bounds improve on the classic $\Omega(\log^* n)$ lower bound by Linial [166], and they also show that Luby's [169] probabilistic $O(\log n)$ time MIS algorithm is close to optimal.

Motivated by the wireless nature of networks like ad hoc and sensor networks, we study local computation in network models that more closely capture *wireless networks* in Chapter 8. In some classes of graphs, we obtain results that exactly capture the amount of local information required to solve a large number of distributed tasks. These results show, for instance, that from the point of view of *locality*, a simple ring network is asymptotically as hard as the vast family of unit ball graphs when the underlying metric space is doubling. Finally, the fact that the doubling dimension of the network's underlying metric space has a significant impact on the running time of distributed algorithms reveals an interesting relationship between *distributed computing* and recent work on *low-dimensional metric spaces*.

Finally, it should be noted that our work on local computation is closely related to—and often goes hand in hand with—another evolving discipline in distributed computing: *distributed approximation*. The study of distributed computing is interesting because it lies on the boundary of well-established areas: *approximation theory* and *distributed computing*.

Chapter 4

Computational Models

As mentioned before, one of the main characteristics of today's most interesting computer networks is their *low coupling level* between the participating entities. Peer-to-peer networks, wireless ad hoc and sensor networks, or the Internet form loosely coupled distributed systems, in which there is restricted global knowledge and no centralized control. Moreover, instead of the tightly synchronized communication via shared memory typically employed by processors of a parallel machine, nodes in multi-hop networks communicate with each other by exchanging messages. In this chapter, we present the classic computational model that explicitly captures the communication between independent entities forming a network: the *message passing model*. This model will be studied throughout Part I of this thesis.

4.1 Message Passing Model in Distributed Computing

In the *message passing model* of distributed computing, the network is modeled as an undirected graph $G = (V, E)$, in which each vertex $v \in V$ represents a node (host, device, processor, ...) of the network. Two nodes $u, v \in V$ are connected by an edge $(u, v) \in E$ if and only if there is a bidirectional communication channel that connects the two nodes. Every node $v \in V$ is assigned a unique identifier $id(v)$ of size $O(\log n)$ bits, which, in reality, may correspond to an IP-address or a MAC-address. A node may communicate directly only with its neighbors in the network graph G, and messages between non-neighboring nodes must be relayed in a multi-hop fashion by nodes on a path connecting the two nodes.

Throughout Part I, we study the *synchronous message passing model*. In this model, all nodes wake up simultaneously and start executing their locally stored distributed algorithm. Communication occurs in discrete, globally synchronized pulses. The time interval between two consecutive pulses is called a *round*. In each round, every node $v \in V$ is allowed to send an arbi-

trary message to each of its neighbors $u \in \Gamma(v)$ in the network graph. Since we consider point-to-point networks, a node may send different messages to different neighbors in the same round. Additionally, every node is allowed to perform local computations based on information obtained in the messages of previous rounds. Communication is reliable, that is, every message that is sent during a communication round is correctly received by the end of the round.

There are two main measures of efficiency of distributed algorithms: the *time complexity* and the *message complexity*. An algorithm's *time complexity* is defined as the number of communication rounds until all nodes terminate. Since the efficiency of a distributed algorithm typically depends mainly on the time used for communication, the model makes no explicit restriction on the amount of local computation. Principally, every node can locally compute an arbitrary computable function in every round. Note, however, that in virtually all algorithms presented in literature, local computation is reasonably small. Unless stated otherwise, our algorithms presented in Part I of this thesis require only minimal local computation. The algorithm's *message complexity* is the total number of messages sent by the nodes in the network throughout the algorithm's execution. For a more formal and thorough introduction to the message passing model of distributed computing and its relevant complexity measures, we refer to standard textbooks, e.g. [16, 196].

The importance of the message passing model stems from both theoretical and practical considerations. From a practical point of view, the model provides a fairly realistic and clean abstraction of many of today's prominent networks, including peer-to-peer networks, wireless networks, or the Internet. From a theoretical point of view, the model is concise and robust enough to allow for rich and profound structural insights into the nature of distributed computing. Moreover, its proximity to other fields of computer science such as graph theory, approximation theory, geometry, or information theory has additionally enriched the literature on this distributed computing model. For these reasons, the message passing model in its various flavors has become the standard model in networking and large areas of distributed computing, e.g. [17, 98, 166, 196, 197].

Depending on the kind of network studied, it may be appropriate to replace our point-to-point network model by a "broadcast" network model. Motivated by wireless radio networks or multi access channels like the Ethernet, the main characteristic of broadcast models is that in a single round, a node cannot send different messages to different nodes. Instead, a message sent by a node may be received by all its neighbors. Also, it is clear that the synchronous message passing model is idealistic in the sense that it abstracts away numerous challenges arising in real networks. In particular, messages exchanged in real networks may be lost or corrupted and nodes may wake up asynchronously, starting the algorithm at different times. For the purpose of studying local computation, however, the model is well suited because it provides a clean abstraction that allows to succinctly analyze the interplay between time complexity, locality, and approximation. In particular, results obtained in the synchronous message passing model capture the fun-

damental *distributed complexity* of network problems without "side-effects" stemming, say, from unreliable communication channels. In Parts II and III of this thesis—when studying the complexity of network coordination tasks from a more low-level perspective—, we will move from the synchronous message passing model to more realistic and harsher computational models, which incorporate additional challenges such as interference, congestion, asynchronous wake-up, or unreliable communication.

Finally, it is important to note that besides the *synchronous* message passing model, research in distributed computing has also prominently featured the *asynchronous* variant of the message passing model. Roughly speaking, this model allows messages to be delayed an arbitrary (but finite) amount of time. The time complexity of an algorithm is the time of a worst-case execution in which—for the sake of analysis—a maximum message delay of 1 time unit is assumed. Throughout this part of the thesis, we deliberately abstract away all problems arising from asynchronous executions. The reason is that when studying local computation, there is a close relationship between an algorithm's time complexity and the locality of the underlying problem. Our main focus being on time complexity and local knowledge, rather than on message complexity, there is a simple method to locally synchronize an asynchronous system. In particular, we can simply employ the so-called α-synchronizer as defined in the work on network synchronizers by Awerbuch [17]. Essentially, this synchronizer lets every node send a message to all its neighbors in every "round", regardless of whether these messages are actually required in a synchronous protocol. A node can proceed to its next round of computation only when it has received the corresponding message from all its neighbors. This procedure guarantees that all nodes are always in the same (or at least subsequent) rounds and the distributed algorithm's execution behaves like in a synchronized system. As shown in [22], the message complexity of the α-synchronizer can be reduced to a polylogarithmic overhead (as compared to the corresponding synchronous algorithm) at the cost of a polylogarithmic deterioration of the time complexity.

There is one aspect that we have only vaguely specified in our description of the message passing model: the messages. In particular, we have made no restriction on the size of the messages, i.e., the number of bits contained in each of them. In fact, it turns out that the question of allowable message size leads to the two fundamental limitations arising in message passing networks: *locality* and *congestion*. The prototypical communication modes that capture these two obstacles have been described in Peleg's seminal book on locality in distributed computing [196]. These models, called the $\mathcal{LOCAL}$ and $\mathcal{CONGEST}$ models in [196], differ in the way they model the amount of information that can be sent in a single message.

4.2 The $\mathcal{LOCAL}$ Model: The Power of Knowing your Neighborhood

The first prototypical model is known as the $\mathcal{LOCAL}$ model [196] or as Linial's free model [166]. In this model, there is no bound on the amount of

information that can be transmitted in a single message, i.e., messages are unbounded.

Definition 4.1. *In the $\mathcal{LOCAL}$ model, every node $v \in V$ can send an arbitrarily large message to every neighbor $u \in \Gamma(v)$ in each round of communication.*

By abstracting away all restricting factors besides locality (e.g., congestion or asynchronity), the $\mathcal{LOCAL}$ model provides an ideal abstraction layer for analyzing the effects of *locality* on distributed computation. In particular, because message size is unlimited, congestion has no influence on an algorithm's time or message complexity. For this reason, the $\mathcal{LOCAL}$ model provides the strongest possible model when proving *lower bounds* on local computation.

The importance of the $\mathcal{LOCAL}$ model also stems from the fact that it provides a one-to-one correspondence between the notion of *time complexity of distributed algorithms* and the graph theoretic notion of *neighborhood-information*. In particular, having a distributed algorithm perform k communication rounds is *equivalent* to a scenario in which distributed decision makers at the nodes of a graph must base their decision on (complete) knowledge about their k-hop neighborhood $\Gamma_k(v)$ only. To see this, note that because messages are unbounded, every node $v \in V$ can collect the identifiers and interconnections of all nodes in its k-hop neighborhood in k communication rounds. Collecting the complete k-neighborhood can be achieved if all nodes send their complete states to all their neighbors in every round. After round i, all nodes know their i-neighborhood. Learning the i-neighborhoods of all neighbors in round $i+1$ suffices to know the $i+1$-neighborhood.

On the other hand, a node cannot obtain any information from a node at distance $k+1$ or more, because sending information over $k+1$ hops requires at least as many communication rounds. Therefore, in the $\mathcal{LOCAL}$ model, knowing one's k-hop neighborhood is exactly as powerful as employing a distributed algorithm with running time k. In this sense, the $\mathcal{LOCAL}$ model concisely relates distributed computation to the *algorithmic theory of the value of information* as studied in [193]: the question of *how much local knowledge* is required for distributed decision makers to solve a global task or approximation a global goal is equivalent to the question of *how many communication rounds* are required by a distributed algorithm to solve the task. This correspondence renders the $\mathcal{LOCAL}$ model an interesting subject of study even beyond distributed computing, but also from a graph- and information-theoretic point of view.

More formally, in k communication rounds, a node v may collect the IDs and interconnections of all nodes in $\Gamma_k(v)$. Each node therefore has a *partial view* of the graph and must base its algorithm's outcome solely on information obtained in this k-hop neighborhood. Formally, let $\mathcal{T}_{v,k}$ be the topology seen by v after these k rounds, i.e. $\mathcal{T}_{v,k}$ is the graph induced by the k-neighborhood of v where edges between nodes at exactly distance k are excluded. The *labelling* (i.e. the assignment of identifiers to nodes) of $\mathcal{T}_{v,k}$ is denoted by $\mathcal{L}(\mathcal{T}_{v,k})$. The view of a node v is the pair $\mathcal{V}_{v,k} := (\mathcal{T}_{v,k}, \mathcal{L}(\mathcal{T}_{v,k}))$. The best a local algorithm with running time k can do in the $\mathcal{LOCAL}$ model

is to gather all information about its k-hop neighborhood and decide upon this information. This shows that in principle, every local algorithm in the $\mathcal{LOCAL}$ model can be transformed into the following canonical form.

1. Collect complete k-neighborhood $\Gamma_k(v)$ in k communication rounds
2. Compute the output by locally simulating the relevant part of the distributed algorithm (no communication required)

It is therefore clear that in the $\mathcal{LOCAL}$ model, every computable problem can be solved in time D, because this is the time required for each node to learn the entire topology of the network graph.

The $\mathcal{LOCAL}$ model was first studied by Linial in his classic paper devoted to the fundamental possibilities and limitations of local computation in graphs [166]. In particular, this paper featured the famous $\Omega(\log^* n)$ time lower bound for computing an MIS on a ring, the only previously known lower bound on local computation that holds even in the $\mathcal{LOCAL}$ model and is thus a true consequence of locality limitations only. The ultimate question concerning local computation in graphs was articulated by Naor and Stockmeyer in their paper entitled 'What Can Be Computed Locally?' [184]. In this paper, the authors prove that there exist locally checkable labelings that can be computed in a constant number of communication rounds. We describe this and other related work on the $\mathcal{LOCAL}$ model when discussing specific network coordination problems in Chapter 5. Chapters 6 and 8 present upper bounds on the distributed complexity of algorithms for various network problems in the $\mathcal{LOCAL}$ model. In combination with the locality lower bounds established in Chapter 7, these results capture the inherent trade-off between the amount of *local knowledge* (or time used by a distributed algorithm) and the *quality of the resulting global solution*.

4.3 The $\mathcal{CONGEST}$ Model: The Impact of Limited Bandwidth

By uniquely focusing on capturing the local nature of a distributed program's execution, the $\mathcal{LOCAL}$ model provides a clean abstraction for describing the locality of distributed computation. On the other hand, the model abstracts away many challenges arising in real world networks, such as congestion and bandwidth restrictions. It is the role of the $\mathcal{CONGEST}$ model [196] to capture these additional aspects of distributed computing. In particular, the $\mathcal{CONGEST}$ model is aimed at incorporating the effects of the *volume* of communication, in addition to its locality.

Definition 4.2. *In the $\mathcal{CONGEST}$ model, every node $v \in V$ can send a message of size $O(\log n)$ bits to every neighbor $u \in \Gamma(v)$ in each round of communication.*

The message size of $O(\log n)$ bits allows each message to contain a constant number of node identifiers and constants that are polynomial in n.

Also, when studying weighted graphs, we typically assume that the weight of an edge is at most polynomial in n and therefore, a weight of a single edge can be communicated in one message.

In the $\mathcal{LOCAL}$ model, every distributed problem can be solved *neighborhood optimally*: every node first collects the entire network topology in time D and solves the problem locally. In the $\mathcal{CONGEST}$ model, however, a node $v \in V$ may be able to gather only a subset of the available information from its neighborhood $\Gamma_k(v)$ due to congestion. Therefore, even problems on a complete graph—which could easily be solved in a single round of communication in the $\mathcal{LOCAL}$ model—become non-trivial in the $\mathcal{CONGEST}$ model. For instance, the fastest known algorithm for the MST problem in a complete graph in the $\mathcal{CONGEST}$ model has a running time of $O(\log \log n)$ [168]. Notice again that the same problem is trivially solved in the $\mathcal{LOCAL}$ model by first exchanging all edge-weights in the graph. Every node then has complete information of the instance and can locally compute the global solution.

In fact, the MST problem has turned out to be of utmost importance in the study of the $\mathcal{CONGEST}$ model and there has been a long line of gradual improvements. The distributed complexity of the MST problem in the $\mathcal{CONGEST}$ was first studied by Gallager, Humblet, and Spira, their algorithm featuring a time complexity of $O(n \log n)$ [98]. Subsequent improvements of this algorithms include [43, 97] and [18], who reduced the time complexity to $O(n \log^* n)$ and $O(n)$, respectively. This last result is "existentially" optimal in the sense that there exist input instances under consideration for which the algorithm is asymptotically best possible.

But what if the network's diameter D is significantly smaller than n? In such networks, it should be possible to obtain even much more efficient algorithms. In particular, considering our focus on the *locality* of distributed computing, it is interesting to precisely identify the parameters of the problem that are inherently responsible for the problem's complexity. In other words, a competitive distributed algorithm for the MST problem should be as local as possible; it should incur little overhead beyond the $\Omega(D)$ time complexity that appears to be inherent. The fundamental question whether there exists such a locality optimal or *neighborhood optimal* algorithm for MST in the $\mathcal{CONGEST}$ model was first raised by Garay, Kutten, and Peleg in [101]. The algorithm presented in [101] has a time complexity of $O(D + n^{0.613} \log^* n)$. This time complexity was later improved to $O(D + \sqrt{n} \log^* n)$ by Kutten and Peleg in [156]. By giving a lower bound of $\Omega(D + \sqrt{n}/\log n)$ in [197], Peleg and Rubinovich not only proved the aforementioned algorithm to be almost optimal, but they managed to more closely quantify the "locality" (or actually, the "non-locality") of the MST problem.

In a recent breakthrough paper, Elkin proved the surprising fact that the diameter D is actually not the parameter that determines the complexity of the MST problem [75]. In particular, Elkin singles out the so-called *MST-radius* $\mu(G)$ of a graph to be the exact parameter that reflects the problems distributed complexity. Interestingly, this parameter can be $n/2$ times smaller than the network's diameter. In particular, [75] presents a randomized algorithm that computes an MST with high probability in time

$O(\mu(G)\cdot\log^3 n+\sqrt{n\log n\log^* n})$. In a remarkable recent work, Elkin not only improved the MST lower bound, but generalized it towards *approximations* of the MST problem [76]. It is shown that every algorithm which computes a tree whose weight is within a factor α of the MST requires at least time $\Omega(\sqrt{n/(\alpha\log n)})$. This implies that for a constant approximation, at least $\Omega(\sqrt{n/\log n})$ rounds are required, thus both improving and generalizing the lower bound of [197].

As mentioned above, the $\mathcal{CONGEST}$ model's limited bandwidth renders solving distributed problems difficult even in graphs with low diameter, in which the problem could easily be solved in the $\mathcal{LOCAL}$ model by simply gathering all information from the entire network. For instance, it has been shown that the MST can be computed in time $O(\log\log n)$ in a complete graph in the $\mathcal{CONGEST}$ model [168], but no corresponding lower bound has been proven. In fact, there appears to be hardly any lower bound on the $\mathcal{CONGEST}$ model for graphs with diameter 1 or 2, and coming up with such lower bounds appears to be challenging because usual arguments based on cuts over which a certain amount of information must flow are ruled out in such graphs. Finding techniques for obtaining such lower bounds is an intriguing open problem.

Chapter 5

Local Network Coordination Problems

There are problems that are inherently non-local in the sense that nodes cannot compute a feasible solution to the problem without gathering information from the entirety of the network, or at least a significant part of it. As seen in Section 4.3, the prototypical non-local problem is the problem of computing a minimum-weight spanning tree (MST). Another instructive example for a non-local problem is the task of two-coloring a ring network. While there exists a beautiful deterministic distributed algorithm by Cole and Vishkin that colors a ring with 3 colors in time $O(\log^* n)$ [58], coloring a ring with an even number of nodes with 2 colors requires at least $n - 1$ communication rounds as formally proven by Linial in [166]. In a proper 2-coloring of a ring, nodes must alternately select colors 1 and 2. Intuitively, this means that every node must obey to a global "sense of direction" in order to decide whether it should take color 1 or 2.

In contrast, many of the important coordination problems in networks appear to be much more local by nature, especially when we allow for non-optimal, approximative solutions to optimization problems. In the sequel, we define such *local network coordination problems* and describe their applications. As we will see, these problems typically boil down to classic graph theoretic constructs.

5.1 Clustering in Ad Hoc and Sensor Networks

The first application scenario is *clustering in wireless multi-hop networks*. Wireless ad hoc and sensor networks consist of autonomous devices communicating via radio. Typically, there is no central server or common, stationary infrastructure which could be used for the organization of the network. Instead, the underlying communication infrastructure has to be provided by the nodes themselves by means of applying distributed algorithms.

Consider the task of routing a message between two distant nodes. In absence of a stationary infrastructure and because the topology of the network may be constantly changing, routing algorithms for ad hoc networks differ significantly from standard routing schemes used in wired networks. One effective way of improving the performance of routing algorithms is to group nodes into *clusters* [11, 216, 227, 229, 238]. The routing is done between clusterheads who act as routers, whereas all other nodes merely communicate via a neighboring clusterhead. Besides facilitating the communication between distant nodes (i.e., routing), another important purpose of clustering in wireless multi-hop networks is to enable efficient communication between adjacent nodes. In particular, many *medium access control (MAC) protocols* are implicitly based on the notion of clusters [122, 203]).

In battery powered *wireless sensor networks* featuring tight energy constraints, structuring the network into energy-efficient clusters plays a key role for prolonging the networks lifetime, e.g., [179, 122]. Only selected clusterleaders remain active, while all other nodes can go into an energy-efficient *sleep mode* thus saving valuable battery power [67].

In all these settings, it is required that every non clusterhead has at least one clusterhead within its communication range. Moreover, from a global point of view, it is advantageous to minimize the number of clusterheads. When modeling the network as a graph $G = (V, E)$, the above clustering problem maps directly to the graph theoretic problem of computing a *minimum dominating set* in a graph.

Definition 5.1 (Minimum Dominating Set (MDS)). *Given a graph $G = (V, E)$. A dominating set is a subset $S \subseteq V$ such that every node is either in S or has at least one neighbor in S. The minimum dominating set problem asks for a dominating set S of minimum cardinality.*

In the centralized, sequential case, the complexity of MDS is understood well. It has been shown in [87, 171] that under reasonable complexity assumptions, the best possible approximation ratio for MDS is $\ln \Delta$. On the other hand, this ratio is achieved by the natural greedy algorithm [54]. Much less is known about the *distributed* approximation of dominating sets. In [156], an algorithm that computes a dominating set of size at most $n/2$ in time $O(\log^* n)$ is presented. While being extremely fast, the algorithm's approximation ratio can be $\Theta(\Delta)$. The algorithm proposed in [238] and its recent adaptation in [65] computes a connected dominating set in constant time, but does not have any worst-case guarantees. In unit disk graphs, the algorithm of [100] computes an O(1)-approximation in expectation and features a running time of $O(\log \log n)$. This result was subsequently generalized in [149]. The first algorithms guaranteeing a non-trivial approximation ratio for general graphs are given in [134] and [202], both achieving an $O(\log \Delta)$ approximation in polylogarithmic time.

The first constant-time algorithm with a non-trivial approximation ratio was given by Kuhn and Wattenhofer in [151]. Their algorithm works in the $\mathcal{CONGEST}$ model and computes an $O(\sqrt{k}\Delta^{1/\sqrt{k}} \log \Delta)$ approximation to MDS in $O(k)$ rounds of communication, for any integer k. While not surpassing the ratios achieved by [134] or [202], the algorithm in [151] is the first

to give a full upper bound on the achievable trade-off between the amount of communication and the resulting global approximation of a local algorithm. In Chapter 6 of this thesis, we improve on these results in the $\mathcal{LOCAL}$ model by giving new approximation guarantees for the generalized facility location problem. Moreover, our hardness of distributed approximation lower bound in Chapter 7 proves that the trade-off achieved by this algorithm is not far from being optimal.

5.2 Covering and Packing Problems

The MDS problem described in Section 5.1 is a well-known *covering problem*. In general, covering problems can be captured as an integer linear program of the following form:

$$\begin{aligned} \min \quad & \underline{c}^T \underline{x} \\ \text{subject to} \quad & A \cdot \underline{x} \geq \underline{b} \\ & x_i \in \mathbb{N}_0. \end{aligned} \qquad (ILP_{Covering})$$

In a covering problem, it is assumed that all entries a_{ij} of A are non-negative and that all entries b_i and c_j are positive. The MDS problem is a covering problem in which all $b_i = c_j = 1$ and the matrix A corresponds to the graph's adjacency matrix. When relaxing the constraints $x_i \in \mathbb{N}_0$ to $x_i \geq 0$, we obtain the linear relaxation of the covering problem, denoted by $LP_{Covering}$. The dual linear program $DLP_{Covering}$ is called a fractional *packing problem* (also denoted by $LP_{Packing}$).

$$\begin{aligned} \max \quad & \underline{b}^T \underline{y} \\ \text{subject to} \quad & A^T \cdot \underline{y} \leq \underline{c} \\ & y_i \geq 0. \end{aligned} \qquad (LP_{Packing})$$

Many important problems can be captured as covering or packing problem. Problems such as (the possibly weighted versions of) minimum set cover or minimum dominating set fall into the category of covering problems. Packing problems occur in a wide range of resource allocation problems. Specific examples include the assignment of flows to a given fixed set of paths or the maximization of a utility in an environment in which a complex combination of possibly scarce resources (bandwidth, memory, CPU power, etc...) must be allocated optimally. Possibly the most simple of all NP-hard covering problems is the *minimum vertex cover* problem.

Definition 5.2 (Minimum Vertex Cover (MVC)). *Given a graph $G = (V, E)$. A vertex cover $S \subseteq V$ is a set of nodes such that for every $e \in E$, at least one incident node is in S. The* minimum vertex cover problem *is to find a vertex cover S of minimum cardinality.*

The minimum vertex cover problem is captured by the following covering integer linear program.

$$\begin{aligned} \min \quad & \sum_{v_i \in V} x_i \\ \text{subject to} \quad & x_i + x_j \geq 1 \qquad ,\forall (v_i, v_j) \in E \\ & x_i \in \{0,1\} \quad ,\forall v_i \in V. \end{aligned} \tag{ILP_{MVC}}$$

The dual packing problem to the ILP_{MVC} integer linear program is the *maximum matching problem* ILP_{MM}. Again, the relaxed fractional versions of both problems are denoted by LP_{MVC} and LP_{MM}, respectively.

$$\begin{aligned} \max \quad & \sum_{e \in E} y_e \\ \text{subject to} \quad & \sum_{e \in E(v)} y_e \leq 1 \qquad ,\forall v \in V \\ & y_e \in \{0,1\} \quad ,\forall e \in E. \end{aligned} \tag{ILP_{MM}}$$

Definition 5.3 (Maximum Matching (MM)). *Given a graph $G = (V, E)$. A matching $M \subseteq E$ is a set of edges, such that no two edges in M have a common endpoint. The maximum matching problem is to find a matching M of maximum cardinality.*

Covering and packing problems are well-studied in the algorithmic literature. Problems such as MVC or MDS appeared in the first list on NP-complete problems presented by Karp [140]. In the sequential, centralized case, the study of linear programming techniques and their use in approximation algorithms for hard combinatorial optimization problems has been at the center of attention in the theoretical computer science community. For a textbook on the role of linear programming techniques in approximation algorithms, we refer for instance to [223]. Sequential algorithms for covering and packing linear programs have been presented in [72, 93, 103, 200], while more general mixed packing and covering linear programs were studied in [94, 141, 241].

In a parallel context, covering and packing linear programs were first studied by Luby and Nisan in [170]. Young presents an efficient parallel algorithm for mixed covering and packing linear programs [241]. Both algorithms are parallel (as opposed to distributed) in the sense that they require global knowledge about the system's state. In [193], Papadimitriou and Yannakakis were the first to study the approximability of a packing linear program in complete absence of any global knowledge, thus initiating the field of *distributed approximation*. The first algorithm with guaranteed constant approximation in polylogarithmic time was given by Bartal, Byers, and Raz in [31]. In particular, the algorithm of [31] computes a $(1+\epsilon)$-approximation in time $O(\log^2(\gamma m)\log(\gamma mn/\epsilon)/\epsilon^3)$, where $\gamma := a_{max}/a_{min}$ is the ratio between the largest and the smallest coefficient in the linear program. Unlike

our parameterized, constant-time algorithms in Chapter 6, however, the algorithm of [31] explores the trade-off between the approximation guarantee and the required locality only partially. In particular, the algorithm has a time complexity of $O(\log^2(\gamma m))$ regardless of the achieved approximation ratio.

In addition to these generic algorithms for general covering and packing linear programs, there are several distributed approximation algorithms for specific optimization problems that fall into the covering/packing category. In Section 5.1, we have already reviewed the relevant literature on the distributed approximation of the minimum dominating set problem. For the *maximum matching* problem, the probabilistic distributed maximal matching algorithms of [9, 129, 169] provide 2-approximations for MM in time $O(\log n)$. Interestingly, the same approximation guarantee can also be obtained deterministically in polylogarithmic time (see Section 5.4) [119, 120]. Also, it was shown that a 1.5 approximation to the maximum matching problem can be computed deterministically in time $O(\log^4 n)$ [64]. Finally, a 5-approximation to the *weighted maximum matching problem* in time $O(\log^2 n)$ was presented in [230].

Intuitively, a simple covering problem such as MVC appears to be ideally suited for *local computation*. In particular, a node should be able to decide whether or not to join the vertex cover by communicating with its neighbors a few times; information about nodes that are very distant appears to have little impact on this decision. Interestingly, as we will shown in Chapter 7, this intuition is misleading and even such a pronounced and seemingly simple *local problems* such as MVC cannot be approximated well in a constant number of communication rounds. In other words, we present *hardness of distributed approximation* lower bounds for MVC and related problems that hold even in the $\mathcal{LOCAL}$ model. This lower bounds the inherent amount of locality (or topological information) needed by distributed decision makers in order to achieve a good approximation to even a simple problem such as MVC.

Section 6.1 presents a distributed algorithm for MVC and for the fractional MM problem. An extended version of an algorithm for general covering and packing linear programs will be presented in Section 6.2. While the MVC and MM algorithms are virtually optimal with regard to the achievable trade-off between locality and approximation ratio, the general upper bound of Section 6.2 is not too far from being optimal either.

5.3 Facility Location

There are important and interesting network coordination problems that appear to have a local characteristic but cannot be captured as a covering and packing pair of linear programs. One of the most studied problems in operations research and the theory of approximation, the *facility location* problem, is such a case. In the facility location problem, there is a set of *clients* (a.k.a. cities or demands) and a set of possible server locations, called *facilities*. Every client must be connected to a facility that serves the client's demand. Opening a facility i causes *opening costs* f_i and connecting a client

j to an opened facility i incurs *connection costs* c_{ij}. The goal is to open a subset of the facilities and connect each client to an opened facility in such a way that minimizes the sum of connection costs and opening costs.

The facility location problem captures a large variety of important application scenarios. Traditionally, it has been used to model the problem of finding the best geographic location for the construction of industrial facilities or warehouses. While this classic application can be satisfactorily solved by a centralized algorithm, there are numerous applications that explicitly demand for *distributed algorithms*. Consider, for instance, the problem of dynamically setting up servers or placing caches in the Internet for a certain application. Setting up a server at a host in the Internet incurs overhead, traffic, and maintenance costs at that particular host. On the other hand, every client wishes to access its data from a server that is as close as possible in order to minimize its delay. The resulting trade-off between the number of servers to be installed and the propagation delay maps precisely to the facility location problem.

Another example for a particularly *distributed* facility location problem is found in battery powered wireless ad hoc and sensor networks. When employing clustering techniques for the purpose of saving energy, the resulting trade-off follows along the same lines. It is desirable to have as few cluster-leaders as possible since this in turn allows more nodes to go into sleep mode (this part is equivalent to the MDS problem). However, having few cluster-heads naturally increases the *distance* between clusterheads and their associated nodes, which forces nodes to set their transmission power to higher values in order to reach their clusterhead. Ideally, a clustering should therefore consist of few clusterheads that are physically close to as many non-leaders as possible.

In the sequel, we formalize a model for the local facility location problem that generalizes the minimum dominating set problem discussed in Section 5.1. In this model, the facility location instance is represented by a bipartite graph $G = (C \cup F, E)$. C and F denote the set of clients and facilities, respectively, and we write $n = |C|$ and $m = |F|$. Each client and facility has a distinct ID of size $O(\log n)$ bits. The non-negative opening costs of facility $v_i \in F$ are denoted by f_i. The connection costs between facility $v_i \in F$ and client $v_j \in C$ are denoted by c_{ij}. Note that we do not assume the connection costs c_{ij} to form a metric and a client can only be connected to a neighboring facility in the graph, i.e., the connection cost c_{ij} is only specified for $v_j \in C$ and $v_i \in F$ such that $v_i \in \Gamma(v_j)$. Also, c_{ij} may be infinitely large. As for notation, we write $F(v_j)$ to denote the set of neighboring facilities of client v_j, and $C(v_i)$ to denote the set of neighboring clients of facility v_i. With these definitions, we can define the local facility location problem as follows.

Definition 5.4 (Facility Location Problem (FL)). *Given a bipartite graph $G = (F \cup C, E)$, where F and C are the sets of* facilities *and* clients*, respectively. The problem is to find a subset $I \subseteq F$ of facilities that should be opened, and a function $\phi : C \to I$ assigning clients to open facilities in such a way that the sum of the opening costs and connection costs are minimized.*

The facility location can be described as an integer linear program ILP_{FL} due to Balinski [26], in which y_i indicates whether facility v_i is opened, and x_{ij} indicates if client v_j is connected to the open facility v_i.

$$\begin{aligned}
\min \quad & \sum_{v_i \in F} f_i y_i \quad + \quad \sum_{v_i \in F} \sum_{v_j \in C} c_{ij} x_{ij} \\
& \sum_{v_i \in F(v_j)} x_{ij} \quad \geq \quad 1 \qquad , \forall v_j \in C \\
& y_i - x_{ij} \quad \geq \quad 0 \qquad , \forall v_j \in C, v_i \in F \\
& x_{ij}, y_i \quad \in \quad \{0,1\} \quad , \forall v_j \in C, v_i \in F.
\end{aligned}$$

The first constraint ensures that each client $v_j \in C$ is assigned to some neighboring facility $v_i \in F(v_j)$. The second constraint guarantees that a client v_j can be assigned only to an open facility v_i. As usual, we obtain the LP-relaxation—denoted by LP_{FL}—by relaxing the integer constraints to $y_i \geq 0$ and $x_{ij} \geq 0$. The relaxed dual program (DLP_{FL}) is:

$$\begin{aligned}
\max \quad & \sum_{v_j \in C} \alpha_j \\
& \alpha_j - \beta_{ij} \quad \leq \quad c_{ij} \qquad , \forall v_j \in C, v_i \in F(v_j) \\
& \sum_{v_j \in C(v_i)} \beta_{ij} \quad \leq \quad f_i \qquad , \forall v_i \in F \\
& \alpha_j, \beta_{ij} \quad \geq \quad 0 \qquad , \forall v_j \in C, v_i \in F.
\end{aligned}$$

Notice that this primal and dual pair of LPs have negative coefficients and do not form a covering-packing pair. Further, observe that this formulation of the facility location problem directly generalizes the *minimum dominating set* (MDS) problem of Section 5.1. Consider an instance of the MDS problem on graph $G = (V, E)$. For every node $u \in V$, define a facility v_i^u and a client v_j^u. Now, connect every client v_j^u to a facility v_i^w if and only if $w \in \Gamma(u)$ in the original graph G. When we now define all facility costs to be 1 and all connection costs as 0, the resulting facility location instance is identical to the original MDS instance. Hence, all results on the local approximability of FL in Chapter 6 (Sections 6.2 and 6.3, in particular) directly carry over to the (weighted) MDS problem as well.

Related Work

Its wide applicability and appealing simplicity have rendered the *uncapacitated facility location problem* one of the most well-studied optimization problems in the literature [26, 61, 125]. It has not only occupied a central place in operations research, but has recently attracted a lot of attention from the perspective of approximation theory [118, 130, 132, 144, 212].

For the general *non-metric case*, Hochbaum [125] showed that the greedy algorithm is an $O(\log n)$ approximation. Set cover being a special case of

facility location, this is asymptotically optimal unless it holds that $NP \subseteq DTIME(n^{O(\log\log n)})$ [87, 171]. The *filtering* technique introduced by Lin and Vitter [165] yields another $O(\log n)$ approximation algorithm. In the metric facility location problem, it is assumed that the connection costs obey the triangle inequality. In that case, the problem remains NP-hard, but constant approximations become possible. The first algorithm achieving a constant approximation ratio was given in [212]. Ever since, a flurry of research activity has lead to various improvements. Also, numerous variants of facility location have been studied, e.g. [118, 220].

Considering the vast literature on the facility location problem, surprisingly little is known about the important *distributed* case. In a seminal paper, Jain and Vazirani [132] claim that their primal-dual algorithm for the metric case of the facility location problem was also suitable in a distributed setting. However, this is only the case if either the instance is a complete-bipartite graph and message-size is unbounded, or the algorithm's time-complexity depends on the size of the problem instance. That is, there is no straightforward distributed implementation of their primal-dual algorithm when restricting the number of communication rounds to an arbitrary constant.

As described in Section 5.2, there have been proposals for approximating covering and packing problems in parallel or in distributed settings. To the best of our knowledge, however, there have so far been no specific results on the distributed approximability of any non-covering or non-packing problem. The results in Sections 6.2 and 6.3 thus push the boundaries of distributed LP approximation.

5.4 MIS and other Exact Problems

So far, we have presented combinatorial *optimization* problems that appear to be "local" in nature. Of particular importance in the context of local computation are certain *exact combinatorial problems*, and specifically the *the maximal independent set* (MIS) problem.

Definition 5.5 (Maximal Independent Set (MIS)). *Given a graph $G = (V, E)$. An independent set is a subset of pair-wise non-adjacent nodes in G. A maximal independent set in G is an independent set $S \subseteq V$ such that for every node $u \notin S$, there is a node $v \in \Gamma(u)$ in S.*

On the one hand, the distributed computing community's interest in the MIS problem stems from its practical importance in various application settings. Specifically, in a network graph consisting of nodes representing processors, an MIS defines a set of processors which can operate in parallel without interference. In wireless ad hoc and sensor networks, for instance, clusterings induced by an MIS have been shown to exhibit particularly desirable properties [11]. Beyond its practical importance, however, the MIS problem is also of outstanding theoretical interest, because it prototypically captures the notion of *symmetry breaking*—one of the central aspects in distributed computing—in a simple, well-defined way. What does symmetry

breaking mean? Clearly, it is trivial to select an MIS in a graph by a sequential algorithm with running time $O(n)$: Greedily pick one node after the other and discard all adjacent nodes of the picked node. While computing an MIS therefore poses no challenge for any centralized algorithm, doing the same in a distributed *local* way is much harder. The reason is that symmetries between nodes must be broken, i.e., of neighboring nodes that appear to be equally qualified to join the MIS, exactly one must be selected.

One way to break symmetries is the use of randomization. And indeed, the fastest known distributed MIS algorithms are elegant probabilistic algorithms with an expected running time of $O(\log n)$ [9, 169]. Although originally proposed for the parallel PRAM model, the algorithms can directly be adapted to both the $\mathcal{LOCAL}$ and $\mathcal{CONGEST}$ models. This shows that in order to compute an MIS, distributed decision makers need to known at most about their $O(\log n)$-local neighborhood.

Breaking symmetries *deterministically* appears to be intrinsically more difficult. For instance, if nodes do not have unique identifiers, it is easy to see that there exists no distributed deterministic algorithm for selecting an MIS, even in a graph consisting of only two nodes. But even if every node has a unique identifier, the problem remains hard. Consider, for instance, the straightforward distributed implementation of the sequential MIS algorithm which works as follows: Every node joins the MIS if it has the smallest ID among its neighbors and if none of its neighbors has already joined the MIS. Unfortunately, this algorithm can result in an entirely sequential execution and linear running time because there may be only a single point of activity at any time.

Interestingly, there exist deterministic distributed algorithms that greatly outperform the aforementioned simple distributed implementation of the greedy algorithm [21, 189]. However, even the fastest currently known solutions fall short of achieving a polylogarithmic time complexity. In fact, the question whether there exists a deterministic distributed algorithm for computing an MIS in general graphs remains one of the well-known open problems in distributed computing [166, 196]. The picture looks different when considering special classes of graphs. On a ring, a rooted tree, or a constant-degree graph, an MIS can be computed in time $O(\log^* n)$ [58, 108]. These algorithms are asymptotically optimal due to a corresponding lower bound of $\Omega(\log^* n)$ for computing an MIS in a ring.

In this thesis, we prove that the complexity of computing an MIS strongly depends on the structural complexity of the underlying graph. In Chapter 7, we prove that in order to solve the MIS problem in general graphs, every node needs to know at least about its $\Omega(\sqrt{\log n / \log\log n})$ neighborhood, which implies a corresponding time lower bound for any distributed algorithm. On the other hand, we show in Chapter 8 that the problem can be solved much more efficiently in vast and practically important classes of graphs (so-called graphs with *bounded independence* and unit ball graphs in which the underlying metric space is doubling). In particular, the problem can be solved deterministically in time $O(\Delta \log^* n)$ (if distances are known to the nodes even in $O(\log^* n)$) in such graphs. In combination with the above

lower bound on general graphs, this establishes a separation result that describes the relative complexity of the underlying graph model with regard to local computability.

In close relation to an MIS is the *maximal matching problem*[1], which is a matching (i.e., a set of edges that do not share a common end-point) that is maximal with regard to inclusion.

Definition 5.6 (Maximal Matching). *Given a graph $G = (V, E)$. A maximal matching in G is a set of non-adjacent edges $M \subseteq E$ such that all edges in $E \setminus M$ have at least one common end-point with an edge in M.*

In contrast to the MIS problem, distributed deterministic algorithms with polylogarithmic running time are known for the maximal matching problem. Such algorithms were given by Hańćkoviak, Karoński, and Panconesi in [119, 120].

5.5 Network Decompositions and other Locality-preserving Structures

With all the different local optimization and exact problems presented in the previous sections, the question is whether there exists some uniting, underlying structural property that captures the essential local nature of these problems. Ideally, there exists a problem-independent method for decomposing a global problem into (globally consistent) locally solvable sub-problems, such that the global solution could be obtained by simply combining the local solutions. One type of network representation that often allows to do exactly this is the concept of *network decompositions*. The idea is to decompose the graph into clusters of small diameter, in which local sub-problems can be efficiently solved in a distributed way. Formally, network decompositions are defined as follows [21].

Definition 5.7 (Network Decomposition). *Given a graph $G = (V, E)$. A* (d, c)-decomposition *is a partition S of G into clusters of diameter at most d such that the cluster graph obtained by contracting each cluster into a single node can be colored using c colors.*

The corresponding notion of a *weak network decomposition* is defined analogously with the only difference that only the clusters' *weak diameter* is bounded by d.

In order to exemplify the usefulness of network decompositions for local distributed computing, consider the distributed computation of an MIS. Given a (d, c)-decomposition, an MIS can be computed as follows. The protocol works in c phases. In the first phase, all clusters with color 1 select an MIS within the cluster and add these nodes to the global MIS. In the

[1] The *maximal matching* problem is not to be confused with the *maximum matching* (MM) problem defined in Section 5.2. While MM represents an optimization problem, the maximal matching problem is an exact problem that can trivially be computed by sequential algorithms.

i'th phase, each cluster with color i computes an MIS within the cluster and adds all those nodes to the global MIS which do not conflict with an MIS node selected by a previously considered cluster. Because the diameter of each cluster is at most d and because clusters with the same color do not have a common edge (and are therefore not interfering), each phase can be computed in $O(d)$ time. Given a (d, c)-decomposition, it is therefore possible to compute an MIS in time $O(c \cdot d)$. In a similar way, network decompositions are useful building blocks for solving a variety of other distributed problems ranging from *distributed coloring* [21] and network synchronization [17] to the computation of sparse spanners [69].

The notion of network decomposition was introduced by Awerbuch, Goldberg, Luby, and Plotkin in [21]. Specifically, [21] proposes a deterministic distributed algorithm for computing a (n^ϵ, n^ϵ)-decomposition in time $O(n^\epsilon)$, where $\epsilon \in O(\sqrt{\log\log n / \log n})$. This result was later improved to $\epsilon \in O(\sqrt{1/\log n})$ by Panconesi and Srinivasan in [189]. This network decomposition algorithm in combination with the above scheme for computing an MIS is currently the fastest known deterministic distributed algorithm for computing an MIS in general graphs. Using a proof technique by Awerbuch and Peleg [23], Linial and Saks showed in [167] that every graph $G = (V, E)$ admits an $(O(k), O(kn^{1/k}))$-decomposition, which can be found by a sequential algorithm. In particular, this implies that every graph can be decomposed in an $(O(\log n), O(\log n))$-decomposition. In addition to this existential result, [167] presents a beautiful randomized distributed algorithm that computes a *weak network decomposition* that essentially matches the optimal bounds. As for deterministic algorithms, Awerbuch, Berger, Cowen, and Peleg show in [20] how to improve the decomposition in [189] and get an $(O(\log n), O(\log n))$-decomposition deterministically in time $O(n^{O(\sqrt{1/\log n})})$ and probabilistically (in combination with [167]) in polylogarithmic time. Finally, an efficient parallel decomposition algorithm was studied in [19].

In a larger context, network decompositions can be regarded as an example of *locality-preserving network representations*, i.e., network representations whose structure succinctly capture certain aspects of the network topology. In his book on locality-sensitive distributed computing, [196], Peleg motivates the use of such methods in distributed computing and describes several locality-preserving representations in detail. Typically, the idea is to decompose the network graph into small-diameter regions that satisfy certain desirable properties, such as low overlap or sparseness. Besides network decompositions, the most important notions of cluster-based representations are *covers* and *partitions*, defined by Awerbuch and Peleg in [17, 23, 194]. Sparse covers and partitions have found countless applications in distributed computing, including for instance network synchronization [17], distributed directories [195], compact routing [4, 24], or the computation of spanners [69]. Representing networks using succinct cluster-based representations such as covers and partitions has found application even beyond distributed computing, including approximation theory and online algorithms. For an overview over many of these results, we refer the reader to Peleg's book [196].

In this thesis, we consider network decompositions in two contexts. First,

we use the decomposition algorithm of [167] to obtain efficient local approximation algorithms for combinatorial optimization problems in Chapter 6.2. In Chapter 8, we then show that a large family of practically important graphs allows for an $(O(1), O(1))$-decomposition which can be computed efficiently even deterministically.

Chapter 6

Local Computation: Upper Bounds

The chapter begins with an upper bound on the most basic covering problem, the minimum vertex cover problem, in Section 6.1. Interestingly, the MVC problem is simple enough to allow deterministic solutions that are as efficient as randomized ones. Our deterministic algorithm in combination with the corresponding lower bound on possibly randomized algorithms in Chapter 7 shows that from the point of view of locality, approximating ILP_{MVC} and its fractional version LP_{MVC} are equally hard. Unfortunately, more sophisticated covering and packing problems do not exhibit the same desirable characteristic, because problems such as MDS or facility location—besides locality—pose the additional challenge of *symmetry breaking*.

As noted in Sections 5.4 and 5.5, symmetry breaking is one of the main challenges in devising local distributed algorithms. Consider for instance the MDS problem on a regular graph. Because every node has the same degree, every node appears to be equally qualified to join the dominating set, but in order to maintain a good approximation ratio only a few nodes are actually allowed to join. In a sequential greedy algorithm, the problem is easily solved by picking "good" (cost-efficient) nodes *consecutively* and updating the remaining node set accordingly. In a local distributed algorithms, on the other hand, nodes cannot employ such an iterative selection procedure and must decide locally (within a few communication rounds) which of these nodes can join.

One way to break symmetries is to employ *randomization*, and indeed, some of the fastest known distributed solutions use randomization, e.g. [134]. Another important tool for dealing with symmetry breaking problems in local algorithms, however, is *LP relaxation*. Intuitively, the idea is to *avoid* breaking the symmetry between equally qualified nodes (for example in the aforementioned MDS example) and instead, let each of these nodes join the dominating set *fractionally*. Technically, the algorithm is divided into two

phases. In the first phase, nodes employ a *deterministic* (or potentially probabilistic) algorithm to solve the linear program relaxation of the combinatorial optimization problem at hand. In the second phase, nodes interpret their fractional values as probabilities and employ a distributed randomized rounding scheme to achieve a feasible integer solution to the original problem. For example, instead of solving MDS or facility location directly, nodes first compute local approximations to the fractional versions of MDS and FL only.

The underlying reason for employing such a two-phase algorithm is that this technique allows to separate the two fundamental challenges arising when devising distributed, local algorithms: *locality* and *symmetry breaking*. The locality nature of problems such as MDS or facility location are captured in their relaxed, fractional LP representation. But, whereas the integer problems of MDS and FL contain the additional complexity of symmetry breaking, these problem's fractional versions purely represent their locality. In other words, locally approximating the LP relaxation allows to capture the underlying combinatorial structure of the problem without the additional difficulty of symmetry breaking, which is completely postponed to the final phase. As it turns out, this scheme has the additional advantage that it allows to establish a complete trade-off between the number of communication rounds k (or in other words, the amount of local knowledge available to the distributed decision makers) and the resulting global approximation ratio, for all integers $k > 0$. This yields non-trivial approximation guarantees even in a constant number of communication rounds.

For the local optimization problems discussed in Chapter 5, *distributed randomized rounding* can be performed in a constant number of communication rounds without incurring too great a loss in approximation quality. The more interesting part when devising local approximation algorithms is therefore to locally approximate the LP relaxation in order to achieve a locality-approximation trade-off for the *fractional versions* of the problems.

As we have seen in Section 5.5, network decomposition is a fundamental tool for solving various distributed tasks in the $\mathcal{LOCAL}$ model. In Section 6.2, we show how—based on an efficient network decomposition algorithm by Linial and Saks [167]—the fractional versions of general covering and packing problems, and even the facility location problem can be approximated locally. In particular, we show that for arbitrary integers k, the fractional facility location problem can be approximated within a factor of $O(m^{1/k})$ in time $O(k)$ in the $\mathcal{LOCAL}$ model.

Unfortunately, this result heavily uses the fact that every node v can collect its entire k-hop neighborhood $\Gamma_k(v)$ in the $\mathcal{LOCAL}$ model. Therefore, the scheme cannot easily be implemented in the $\mathcal{CONGEST}$ model with limited bandwidth and therefore, we propose an approach with a different flavor for this model. In the sequential, centralized case, most of the local problems discussed in Chapter 5 have simple *greedy algorithms* that achieve asymptotically optimal approximation guarantees. The idea for obtaining distributed approximation algorithms in the $\mathcal{CONGEST}$ model is therefore to implement an efficient version of a *distributed greedy algorithm*. This algorithm, as well

as a discussion of the difficulties arising in the parallelization of greedy algorithms is given in Section 6.3. Naturally, the time-approximation trade-off achieved by this algorithm is somewhat worse as compared to the algorithm for the $\mathcal{LOCAL}$ model.

Finally, we are not only interested in achieving solutions to fractional problems, but we also want to end up obtaining feasible solutions to the original (integer) optimization problems. For this purpose, we present a *distributed randomized rounding* procedure for the facility location problem in Section 6.4. In combination with the local approximation algorithms for the fractional facility location problem in Sections 6.2 and 6.3, the distributed randomized rounding step produces local algorithms for the facility location problem.

6.1 Minimum Vertex Cover: A Simple Distributed Approximation Algorithm

The MVC problem appears to be an ideal starting point for studying the power of local approximation algorithms. In particular, MVC does not involve the aspect of *symmetry breaking* which is so crucial in more complex combinatorial problems: A fractional solution to LP_{MVC} can be turned into an integer solution to MVC by rounding up all nodes with a fractional value at least 1/2. This increases the approximation ratio by at most a factor of 2. Moreover, any maximal matching is a 2-approximation for MVC and hence, the randomized parallel algorithm for maximal matching by Israeli et al. provides a 2-approximation in time $O(\log n)$ with high probability [129]. This indicates that the amount of locality required in order to achieve a constant approximation for MVC is bounded by $O(\log n)$. In this section, we present a simple distributed algorithm that places an upper bound on the achievable trade-off between time-complexity and approximation ratio for the minimum vertex cover problem.

Specifically, the algorithm comes with a parameter k, which can be any integer larger than 0. The algorithm's time complexity—and hence its locality—is $O(k)$ and its approximation ratio depends inversely on k. The larger k, the better the achieved global approximation.

Algorithm 6.1 simultaneously approximates both MVC and its dual problem, the fractional maximum matching problem LP_{MM}. Let E_i denote the set of incident edges of node v_i. The idea is to compute a feasible solution for minimum vertex cover (MVC) and while doing so, distribute dual values y_j among the incident edges of each node. Each node v_i that joins the vertex cover S sets its x_i to 1 and subsequently, the sum of the dual values y_j of incident edges $e_j \in E_i$ is increased by 1 as well. Hence, at the end of each iteration of the main loop, the invariant $\sum_{v_i \in V} x_i = \sum_{e_j \in E} y_j$ holds. We will show that for all nodes v_i, $\sum_{e_j \in E_i} y_j \leq \alpha$ for $\alpha = 3 + \Delta^{1/k}$ and that consequently, dividing all y_j by α yields a feasible solution for LP_{MM}. By LP duality, α is an upper bound on the approximation ratio for FMM and MVC. We call an edge *covered* if at least one of its endpoints has joined the

Code executed by node $v_i \in V$

1: $x_i := 0$; $\forall e_j \in E_i : y_j := 0$;
2: **for** $\ell := k-1$ to 0 by -1 **do**
3: $\quad \tilde{\delta}_i := |\{\text{uncovered edges } e \in E_i\}| = |\tilde{E}_i|$;
4: $\quad \tilde{\delta}_i^{(1)} := \max_{i' \in \Gamma(v_i)} \tilde{\delta}_{i'}$;
5: $\quad$ **if** $\tilde{\delta}_i \geq (\tilde{\delta}_i^{(1)})^{\ell/(\ell+1)}$ **then**
6: $\quad\quad x_i := 1$;
7: $\quad\quad \forall e_j \in \tilde{E}_i : y_j := y_j + 1/\tilde{\delta}_i$;
8: $\quad$ **end if**
9: $\quad Y_i := \sum_{e_j \in E_i} y_j$;
10: $\quad$ **if** $(x_i = 0)$ and $(Y_i \geq 1)$ **then**
11: $\quad\quad x_i := 1$;
12: $\quad\quad \forall e_j \in E_i : y_j := y_j \cdot (1 + 1/Y_i)$;
13: $\quad$ **end if**
14: **end for**
15: $Y_i := \sum_{e_j \in E_i} y_j$;
16: $\forall e_j = (v_i, v_{i'}) \in E_i : y_j := y_j / \max\{Y_i, Y_{i'}\}$;

Algorithm 6.1: MVC-FMM-Algorithm

vertex cover. The set of uncovered edges incident to a node v_i is denoted by $\tilde{E}_i$, and we define node v_i's *dynamic degree* to be $\tilde{\delta}_i := |\tilde{E}_i|$. The maximum dynamic degree $\tilde{\delta}_{i'}$ among all neighbors $v_{i'}$ of v_i is denoted by $\tilde{\delta}_i^{(1)}$.

In the algorithm, a node joins the vertex cover if it has a large dynamic degree—i.e., many uncovered incident edges—relative to its neighbors. In this sense, it is a faithful distributed implementation of the natural sequential greedy algorithm. Because the running time is limited to k communication rounds, however, the greedy selection step must inherently be parallelized, even at the cost of sub-optimal decisions.

The following lemma bounds the resulting decrease of the maximal dynamic degree in the network.

Lemma 6.1. *At the beginning of each iteration, it holds that* $\tilde{\delta}_i \leq \Delta^{(\ell+1)/k}$ *for every* $v_i \in V$.

Proof. The proof is by induction over the main loop's iterations. For $\ell = k-1$, the lemma follows from the definition of Δ. For subsequent iterations, we show that all nodes having $\tilde{\delta}_i \geq \Delta^{\ell/k}$ set $x_i := 1$ in Line 6. In the algorithm, all nodes with $\tilde{\delta}_i \geq (\tilde{\delta}_i^{(1)})^{\ell/(\ell+1)}$ set $x_i := 1$. Hence, we have to show that for all v_i, $(\tilde{\delta}_i^{(1)})^{\ell/(\ell+1)} \leq \Delta^{\ell/k}$. By the induction hypothesis, we know that $\tilde{\delta}_i \leq \Delta^{(\ell+1)/k}$ at the beginning of the loop. Since $\tilde{\delta}_i^{(1)}$ represents the dynamic degree $\tilde{\delta}_{i'}$ of some node $v_{i'} \in \Gamma(v_i)$, it holds that $\tilde{\delta}_i^{(1)} \leq \Delta^{(\ell+1)/k}$ for every such v_i and the claim follows because $(\tilde{\delta}_i^{(1)})^{\ell/(\ell+1)} \leq \Delta^{\frac{\ell+1}{k} \cdot \frac{\ell}{\ell+1}}$. □

The next lemma bounds the sum of dual y values in E_i for an arbitrary node $v_i \in V$. For that purpose, we define $Y_i := \sum_{e_j \in E_i} y_j$.

Lemma 6.2. *At the end of the algorithm, for all nodes $v_i \in V$,*

$$Y_i \;=\; \sum_{e_j \in E_i} y_j \;\leq\; 3 + \Delta^{1/k}.$$

Proof. Let Φ_h denote the iteration in which $\ell = h$. We distinguish three cases, depending on whether (or in which line) a node v_i joins the vertex cover. First, consider a node v_i which does not join the vertex cover. Until Φ_0, it holds that $Y_i < 1$ since otherwise, v_i would have set $x_i := 1$ in Line 11 of a previous iteration. In Φ_0, it must hold that $\tilde{\delta}_i = 0$ because all nodes with $\tilde{\delta}_i \geq 1$ set $x_i := 1$ in the last iteration. That is, all adjacent nodes $v_{i'}$ of v_i have set $x_{i'} := 1$ before the last iteration and Y_i does not change anymore. Hence, $Y_i < 1$ for nodes which do not belong to the vertex cover constructed by the algorithm.

Next, consider a node v_i that joins the vertex cover in Line 6 of an arbitrary iteration Φ_ℓ. With the same argument as above, we know that $Y_i < 1$ at the beginning of Φ_ℓ. When v_i sets $x_i := 1$, Y_i increases by one. In the same iteration, however, neighboring nodes $v_{i'} \in \Gamma(v_i)$ may also join the vertex cover and thereby further increase Y_i. By the condition in Line 5, those nodes have a dynamic degree at least $\tilde{\delta}_{i'} \geq (\tilde{\delta}_{i'}^{(1)})^{\ell/(\ell+1)} \geq \tilde{\delta}_i^{\ell/(\ell+1)}$. Further, it holds by Lemma 6.1 that $\tilde{\delta}_i \leq \Delta^{(\ell+1)/k}$ and therefore

$$\tilde{\delta}_i \cdot \frac{1}{\tilde{\delta}_{i'}} \;\leq\; \frac{\tilde{\delta}_i}{\tilde{\delta}_i^{\ell/(\ell+1)}} \;=\; \tilde{\delta}_i^{1/(\ell+1)} \;\leq\; \Delta^{1/k}.$$

Thus, edges that are simultaneously covered by neighboring nodes may entail an additional increase of Y_i by $\Delta^{1/k}$. Together with v_i's own cost of 1 when joining the vertex cover, the total increase of Y_i in Line 7 of Φ_ℓ is then at most $1 + \Delta^{1/k}$. In Line 7, dual values are distributed among uncovered edges only. Therefore, the only way Y_i can increase in subsequent iterations is when neighboring nodes $x_{i'}$ set $x_{i'} := 1$ in Line 11. The sum of the y_j of all those edges covered only by v_i (note that only these edges are eligible to be increased in this way) is at most 1. In Line 12, these y_j can be at most doubled. Putting everything together, we have $Y_i \leq 3 + \Delta^{1/k}$ for nodes joining the vertex cover in Line 6.

Finally, we study nodes v_i that join the vertex cover in Line 11 of some iteration Φ_ℓ. Again, it holds that $Y_i < 1$ at the outset of Φ_ℓ. Further, using an analogous argument as above, Y_i is increased by at most $\Delta^{1/k}$ due to neighboring nodes joining the vertex cover in Line 6 of Φ_ℓ. Through the joining of v_i, Y_i further increases by no more than 1. Because the y_j are increased proportionally, no further increase of Y_i is possible. Thus, in this case we have $Y_i \leq 2 + \Delta^{1/k}$. □

Based on the bound obtained in Lemma 6.2, the main theorem follows from LP duality.

Theorem 6.3. *In k rounds of communication, Algorithm 6.1 achieves an approximation ratio of $O(\Delta^{1/k})$ even in the $\mathcal{CONGEST}$ model. The algorithm is deterministic and requires $O(\log \Delta)$ and $O(\log \Delta / \log\log \Delta)$ rounds for a constant and polylogarithmic approximation, respectively.*

Proof. We first prove that the algorithm computes feasible solutions for MVC and fractional maximum matching. For MVC, this is clear because in the last iteration, all nodes having $\tilde{\delta}_i \geq 1$ set $x_i := 1$. The dual y-values form a feasible solution because in Line 16, the y_j of each edge e_j is divided by the larger of the Y_i of the two incident nodes corresponding to e_j, and hence, the constraint of LP_{MM} is guaranteed to be satisfied. The algorithm's running time is $O(k)$, because every iteration can be implemented with a constant number of communication rounds even in the $\mathcal{CONGEST}$ model. As for the approximation ratio, it follows from Lemma 6.2 that each y_j is divided by at most $\alpha = 3 + \Delta^{1/k}$ and therefore, the objective functions of the primal and the dual problem differ by at most a factor α. By LP duality, α is a bound on the approximation ratio for both problems. Finally, setting $k_1 = \beta \log \Delta$ and $k_2 = \beta \log \Delta / \log\log \Delta$ for an appropriate constant β leads to a constant and polylogarithmic approximation ratio, respectively. □

The reason why Algorithm 6.1 does not achieve a constant or polylogarithmic approximation ratio in a constant number of communication rounds is that it "discretizes" the greedy step. Whereas the sequential greedy algorithm would select a single node with maximum dynamic degree in each step, a k-local distributed algorithm must inherently take many such decisions in parallel. This discrepancy between Algorithm 6.1 and the simple sequential greedy algorithm can be seen even in simple networks. Consider for instance the network induced by the complete bipartite graph $K_{m,\sqrt{m}}$. When running the algorithm with parameter $k = 2$, it holds for every node v_i that $\tilde{\delta}_i \geq (\tilde{\delta}_i^{(1)})^{\ell/(\ell+1)}$ in the first iteration ($\ell = 1$) of the loop. Hence, every node will join the vertex cover, resulting in a cover of cardinality $m+\sqrt{m}$. The optimal solution being $\sqrt{m}$, the resulting approximation factor is $\sqrt{m} + 1 = \Delta^{1/2} + 1$.

Intuitively, it seems plausible that better distributed approximation algorithms than Algorithm 6.1 exist. After all, MVC appears to be a particularly local problem and distant nodes do not seem to influence a node's decision. As we prove in Chapter 7, however, the trade-off achieved by Algorithm 6.1 is essentially optimal for any (possibly randomized) distributed algorithm. Hence, there exists no distributed approximation algorithm with significantly better (asymptotic) guarantees, and the amount of locality required by Algorithm 6.1 to solve MVC (as well as fractional maximum matching) is inherent.

Algorithm 6.1 is deterministic and computes a valid solution for ILP_{MVC}, not only for its fractional version. As argued in Section 5.4, one of the major challenges arising in the design of distributed (and particularly local) algorithms is to efficiently break symmetries. Breaking these symmetries in a *deterministic* way has proven to be particularly difficult and there are

currently no deterministic, polylogarithmic distributed algorithms for characteristic symmetry breaking problems such as MIS or network decomposition. In this context, it is interesting to note that because of its particularly simple combinatorial structure, MVC allows for efficient (polylogarithmic) *deterministic* distributed algorithms.

6.2 Facility Location: Distributed Approximation in the $\mathcal{LOCAL}$ Model

In the previous section, we have seen how simple covering and packing problems can be approximated in a local, distributed fashion. The question is, whether a similar approach may be used for approximating other problems as well. Unfortunately, it appears that achieving non-trivial approximation guarantees for more general covering and packing problems (such as MDS, for instance) or the facility location problem requires more sophisticated techniques.

As argued in Section 5.5, network decompositions are fundamental to computation in the $\mathcal{LOCAL}$ model in the sense that they can be applied to obtain fast algorithms for a large class of problems. And indeed, network decompositions can be employed as a building block for locally approximating a wide range of combinatorial optimization problems, including general covering and packing problems, or even the facility location problem. In [167], Linial and Saks presented a randomized distributed algorithm to decompose a graph into sub-graphs of limited diameter. We use their algorithm to decompose the linear program into sub-programs which can be solved locally in the $\mathcal{LOCAL}$ model. In [150], we have shown how to use this technique for solving the fractional version of general covering and packing problems. In this section, we apply the method to the even more general facility location problem, which—as shown in Section 5.3—does not form a covering LP.

Intuitively, the output of a single iteration of the algorithm in [167] consists of connected components of G with the following properties.

1. Different components are far enough from each other such that a local linear program for each component can be defined in a way in which the LPs of any two components do not interfere.

2. Each node belongs to one of the components with probability at least p, where p depends on the diameter the components are allowed to have.

Because of the limited diameter, the LPs of each component can then be computed locally. We then apply the decomposition process in parallel to ensure that with high probability each client is connected to an opened facility a logarithmic number of times.

More specifically, the algorithm by Linial and Saks [167] iteratively constructs the network decomposition one color-class at a time. For each such color class and for parameters B and p, the algorithm yields a subset $\mathcal{S} \subset V$

of the nodes, such that each node $v \in S$ has a *leader* $C(v) \in V$ and the following properties hold:

1. $\forall u \in \mathcal{S}:\ \mathrm{d}(u, C(u)) < B$
2. $\mathcal{S}$ forms a *2-separated partial partition* [196]. That is, it holds for any pair of nodes $u, v \in \mathcal{S}$ with $C(u) \neq C(v)$ that $d(u, v) \geq 2$.
3. $\forall u \in V:\ Pr[u \in \mathcal{S}] \geq p(1-p^B)^n$.

Further, the running time required for computing $\mathcal{S}$ is B. In Algorithm 6.2, we apply the above scheme many times in parallel to an auxiliary graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, setting $B := k$ and $p := 1/m^{1/k}$. The solution $\mathcal{S}^\ell$ constructed in every such independent instance ℓ of the decomposition features the following properties.

i) $\forall u \in \mathcal{S}^\ell:\ \mathrm{d}(u, C(u)) < k$

ii) $\forall u, v \in \mathcal{S}^\ell:\ C(u) \neq C(v) \implies (u, v) \notin \mathcal{E}$.

iii) $\forall u \in \mathcal{V}:\ Pr[u \in \mathcal{S}^\ell] \geq \frac{1}{em^{1/k}}$.

iv) $\mathcal{S}^\ell$ can be computed in k communication rounds.

Let $G = (V, E)$ be the bipartite facility location graph consisting of nodes $V = (C \cup F)$ and consider the linear program relaxation LP_{FL} and its dual DLP_{FL} as defined in Section 5.3. When specifically distinguishing client and facility nodes, we use the notation v_j and v_i, respectively, and recall that $n := |C|$ and $m := |F|$. For the decomposition obtained in the algorithm, we define local linear programs LP^ℓ_{FL} and DLP^ℓ_{FL} which are induced by all facilities and clients that have the same leader. In the sequel, we call the set of facility and client nodes with the same leader to be an *LP-component*.

Let $R \in V$ be an LP-component of nodes with the property that all border-nodes of R (i.e., all nodes $v \in R$ for which $\Gamma(v) \cap (V \setminus R) \neq \emptyset$) are facilities. Let OPT be the objective value of the optimal solution to the facility location problem LP_{FL}. The following lemma captures a straightforward relation between the LPs induced by R and the original LPs LP_{FL} and DLP_{FL}.

Lemma 6.4. *In the optimal solution to the LP induced by R, every client $v_j \in R \cap C$ is connected to an open facility $v_i \in R \cap F$. Furthermore, the dual solution is feasible for DLP^ℓ_{FL}, i.e.,*

$$\forall v_i \in R: \qquad \sum_{v_j \in \Gamma(v_i)} \beta_{ij} \leq f_i$$

$$\forall v_j \in R, v_i \in \Gamma(v_j): \qquad \alpha_j - \beta_{ij} \leq c_{ij}.$$

Finally, it holds that the objective value of the problem induced by R is at most OPT, formally, $\sum_{v_i \in R} f_i y_i + \sum_{v_i \in R} \sum_{v_j \in R} c_{ij} x_{ij} \leq OPT$.

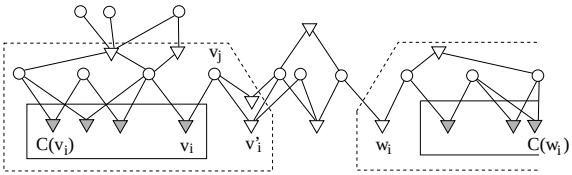

Figure 6.1: A possible outcome of the decomposition algorithm. Facilities and clients are represented as triangles and circles, respectively. Shaded facilities in the solid black rectangle have been selected to be in S^ℓ by the decomposition algorithm. Nodes in the dashed rectangle are added to S^ℓ by Algorithm 6.2 and represent LP-components. Notice that no node occurs in a constraint of more than one LP-component.

Proof. The first claim is clear from the definition of the LP induced by R. The dual feasibility and the upper bound on the objective value follow from the fact that every border node of R is a facility. That is, every client in R has all its neighboring facilities also in R. Hence, a feasible dual solution for the LP induced by R must also be feasible for LP_{FL}. Moreover, the optimal solution for LP_{FL} must connect every client in R to facilities in R and the objective value for the LP induced by R is therefore bounded by OPT. □

We want to argue that the different LP-components constituting a set S^ℓ of every parallel instance of the decomposition algorithm can be solved completely independently from each other. Unfortunately, when running the decomposition algorithm on the original graph G, this independence is not guaranteed because, for instance, a facility can have neighboring clients in two different LP-components. In order to achieve the desired degree of separation between LP-components, we let the nodes run the entire algorithm on the auxiliary graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, whose node set is the set of facilities, $\mathcal{V} = F$, and there is an edge between two facilities if their distance in the original graph is 6 or less,

$$\mathcal{E} := \{(v_i, v_i') | v_i, v_i' \in \mathcal{V} \wedge d_G(v_i, v_i') \leq 6\}.$$

By this, we can guarantee that non-adjacent facilities in $\mathcal{G}$ have at least a distance of 8 hops in the original graph G. Further, a message over an imaginary edge of $\mathcal{G}$ can be sent in 6 communication rounds on the network graph G. In other words, executing the decomposition algorithm on $\mathcal{G}$ instead of G increases the time complexity by at most a factor 6.

The algorithm now proceeds as shown in Algorithm 6.2. Each facility simulates $N = 5em^{1/k} \ln(n+m)$ independent executions of the decomposition algorithm on $\mathcal{G}$ in parallel. This is possible because in the $\mathcal{LOCAL}$ model, message size is unbounded and the messages of the different instances can be sent in a single message. If in such an instance, a facility v_i is in S^ℓ, all its neighboring clients $v_j \in \Gamma(v_i)$ also join S^ℓ. Because we want all border-nodes of LP-components to be facilities, all neighboring facilities of

Code executed by node $v \in V$:

1: Run $N = 5em^{1/k} \ln(n+m)$ instances of GRAPH_DECOMP on $\mathcal{G}$ in parallel;
2: **if** $v = v_j \in C$ **then**
3: For each facility $i \in \Gamma(v_j)$ do $x_{ij} := (\sum_{\ell: v_j \in \mathcal{S}^\ell} x_{ij}^\ell)/\ln(n+m)$;
4: **else if** $v = v_i \in F$ **then**
5: $y_i := (\sum_{\ell: v_i \in \mathcal{S}^\ell} y_i^\ell)/\ln(n+m)$;
6: **end if**;

Procedure GRAPH_DECOMP

1: Run graph decomposition of [167] on $\mathcal{G}$ with $B := k$ and $p := 1/m^{1/k}$;
2: **if** $v_j \in C$ and some neighboring facility $v_i \in \Gamma(v_j)$ is in $\mathcal{S}^\ell$ **then**
3: v_j joins $\mathcal{S}^\ell$ with $C(v_j) := C(v_i)$;
4: **end if**;
5: **if** $v_i \in F \setminus \mathcal{S}^\ell$ and some neighboring client $v_j \in \Gamma(v_i)$ is in $\mathcal{S}^\ell$ **then**
6: v_i joins $\mathcal{S}^\ell$ with $C(v_i) := C(v_j)$;.
7: **end if**;
8: **if** $v_i \in \mathcal{S}^\ell \cap F$ **then**
9: **send** f_i and all connection costs c_{ij} for $v_j \in \Gamma(v_i)$ to $C(v_i)$.
10: **end if**;
11: **if** $v = C(u)$ for some $u \in \mathcal{S}^\ell$ **then**
12: locally compute optimal feasible solutions for LP_{FL}^ℓ and DLP_{FL}^ℓ induced by all facilities and clients $u \in \mathcal{S}^\ell$ for which $v = C(u)$.
13: **send** resulting values x_{ij}^ℓ, y_i^ℓ, α_j^ℓ, and β_{ij}^ℓ to nodes holding the respective variables.
14: **end if**

Algorithm 6.2: Facility Location Algorithm in $\mathcal{LOCAL}$ model

clients in S^ℓ also join S^ℓ. Note that by the definition of $\mathcal{G}$ and by Property (ii) of the decomposition algorithm, two nodes v and v' belonging to different LP-components (i.e., that have a different leader) have distance at least $d(v,v') \geq 4$ in G (see Figure 6.1). Hence, it follows that the leader assignment in Lines 3 and 6 of Algorithm 6.2 is always well-defined. Moreover, the LP-components are independent in the sense that there exists no variable in either LP_{FL} or DLP_{FL}, which appears in a constraints induced by two different LP-components. Therefore, computing an optimal solution for each LP-component independently is valid and the properties of Lemma 6.4 apply. At the end of the algorithm, the variables y_i^ℓ and x_{ij}^ℓ of the N sub-LPs S^ℓ are added up at every node and divided by $\ln(n+m)$.

Theorem 6.5. *Algorithm 6.2 has a running time of $O(k)$ rounds. With high probability, it computes a feasible solution to the fractional facility location problem and achieves an approximation ratio of $O(m^{1/k})$, where m is the number of facilities.*

Proof. We start with the running time. The N executions can be performed

completely in parallel in the $\mathcal{LOCAL}$ model and we therefore focus on one instance of the basic algorithm. By Property (i), collecting the topology of the LP-component and the subsequent distribution of the variables (Lines 6,8,12) can be performed in $O(k)$ time. The same holds for the graph decomposition in Line 1 by Property (iv).

For the approximation ratio, we have to bound the ratio between the primal and the dual objective value at the end of the algorithm. Let $X_{ij} := \sum_{\ell:v_j \in S^\ell} x_{ij}^\ell$ denote the sum of the primal variables x_{ij}^ℓ over all N parallel executions of the decomposition algorithm and let Y_i, $\mathcal{A}_j$, and $\mathcal{B}_{ij}$ be the analogous sums for the other primal and dual variables. In each LP-component, the leader computes an optimal solution to the sub-LP induced by the LP-component, and therefore the primal and dual objective value are equal within an LP-component, i.e., $\sum_{v_i \in S^\ell} f_i y_i^\ell + \sum_{v_i \in S^\ell} \sum_{v_j \in V} c_{ij} x_{ij}^\ell = \sum_{v_j \in S^\ell} \alpha_j^\ell$. When summing up all primal and dual variables at the end of the algorithm, it therefore holds that

$$\sum_{v_i \in V} f_i Y_i + \sum_{v_i \in V} \sum_{v_j \in V} c_{ij} X_{ij} \;=\; \sum_{v_j \in V} \mathcal{A}_j. \tag{6.1}$$

Unfortunately, the summed up dual variables $\mathcal{A}_j$ and $\mathcal{B}_j$ may no longer constitute a feasible solution to the dual problem DLP_{FL}. In order to bound the degree of infeasibility of the dual variables, we note that by Lemma 6.4, the dual variables of each of the N sub-LPs constitute a feasible solution for DLP_{FL}. Because at most N values are added up, it holds that

$$\begin{aligned} \forall v_i \in V: \quad & \sum_{v_j \in \Gamma(v_i)} \mathcal{B}_{ij} \;\leq\; N \cdot f_i \\ \forall v_j \in V, v_i \in \Gamma(v_j): \quad & \mathcal{A}_j - \mathcal{B}_{ij} \;\leq\; N \cdot c_{ij}. \end{aligned}$$

We can therefore obtain a feasible dual solution by setting $\alpha_j := \mathcal{A}_j/N$ and $\beta_{ij} := \mathcal{B}_{ij}/N$.

The final primal variables y_i and x_{ij} are set in the algorithm as $x_{ij} := X_{ij}/\ln(n+m)$ and $y_i := Y_i/\ln(n+m)$, respectively. We now show that these variables satisfy all constraints of LP_{FL} with high probability. By Lemma 6.4, each time a client occurs in some S^ℓ, its primal constraints are satisfied, i.e., it is completely connected to (fractionally) open facilities. With high probability, each client occurs in no fewer than $\ln(n+m)$ different sets S^ℓ: A client joins S^ℓ whenever at least one of its neighboring facilities is in S^ℓ. By Property (iv) of the network decomposition algorithm, the probability of a facility node being in S^ℓ is at least $1/(em^{1/k})$ in each of the N independent instances. Let Z denote the number of times, a client is selected to be in S^ℓ by the graph decomposition algorithm. Then, by Chernoff bound,

$$\begin{aligned} Pr[Z < \ln(n+m)] \;&<\; \left(\frac{e^{-4/5}}{(1-4/5)^{(1-4/5)}}\right)^{5\ln(n+m)} \;=\; \frac{e^{\ln 5 \cdot \ln(n+m)}}{e^{4\ln(n+m)}} \\ &=\; \frac{1}{(n+m)^{4-\ln 5}} \;<\; \frac{1}{(n+m)^2}. \end{aligned}$$

Hence, with probability at least $1-1/(n+m)$ it holds for every client $v_j \in V$ and every facility $v_i \in V$ that

$$\begin{aligned} \forall v_j \in V: \quad \sum_{v_i \in \Gamma(v_j)} X_{ij} &\geq \ln(n+m) \\ \forall v_i \in V, v_j \in \Gamma(v_i): \quad Y_i - X_{ij} &\geq 0. \end{aligned}$$

It follows that with high probability, the variables $x_{ij} := X_{ij}/\ln(n+m)$ and $y_i := Y_i/\ln(n+m)$ as computed in Algorithm 6.2 form a feasible solution to LP_{FL}.

The approximation ratio γ now follows from LP duality, Equality (6.1), and the fact that

$$\sum_{v_i \in V} f_i \cdot \frac{Y_i}{\ln(n+m)} + \sum_{v_i \in V} \sum_{v_j \in V} c_{ij} \cdot \frac{X_{ij}}{\ln(n+m)} = \gamma \cdot \sum_{v_j \in V} \frac{\mathcal{A}_j}{N},$$

for $\gamma = N/\ln(n+m) \in O(m^{1/k})$. □

Setting $k = O(\log m)$, we obtain the following corollary that places an upper bound on the amount of time and locality required for a constant approximation.

Corollary 6.6. *Algorithm 6.2 computes a constant approximation to LP_{FL} in time $O(\log m)$ communication rounds.*

Distributed Covering and Packing Problems The facility location algorithm presented in this Section is analogous to our algorithm for the distributed approximation of general *covering and packing linear programs* in [150]. In particular, we have shown in [150] that based on the decomposition algorithm by Linial and Saks [167], every *covering and packing linear program* can be approximated within a factor of $O(n^{1/k})$ in $O(k)$ communication rounds in the $\mathcal{LOCAL}$ model. Here n is the number of primal variables, for instance the number of nodes in the MDS problem. In other words, in order to achieve a global $O(n^{1/k})$ approximation to a covering or packing linear program in a distributed setting, each node $v \in V$ must have knowledge only about its $O(k)$-hop neighborhood $\Gamma_{O(k)}(v)$ in the graph. This gives an answer to a question raised by Papadimitriou and Yannakakis in [193].

6.3 Facility Location: Distributed Approximation in the $\mathcal{CONGEST}$ Model

The algorithm in the previous section highlights the trade-off between approximation and time complexity of local computation. On the other hand, the algorithm makes heavy use of the fact that in the $\mathcal{LOCAL}$ model, messages can be unbounded. The question is whether similar trade-offs can be achieved even if the exchangeable *volume of information* is limited and

nodes cannot simply gather the entire topological information about their k-hop neighborhood in k communication rounds. For several reasons, there appears to be no simple implementation of the algorithm in Section 6.2 in the $\mathcal{CONGEST}$ model without overly deteriorating its running time. Therefore, we resort to a different idea.

In the MVC algorithm of Section 6.1, the idea was to imitate the sequential greedy algorithm and parallelize its greedy step. Since greedy algorithms are known to yield asymptotically optimal approximation guarantees for problems like MDS and facility location (at least in its general, non-metric case) [54, 125], the question is whether there exist efficient distributed local implementations of these algorithms. In the case of the FL problem, roughly speaking, the sequential greedy algorithm iteratively opens the *star* (consisting of a facility and several neighboring clients) with the best *cost efficiency*. A simple distributed version of this process can be implemented based on the following observation: A facility's cost efficiency can only be reduced if a neighboring client connects to a facility at distance 2. Therefore, if the cost efficiency of a facility v_f is greater than the cost efficiency of any other facility in $\Gamma_2(v_f)$, the sequential greedy algorithm opens v_f and assigns the corresponding clients of the star before any other facility in $\Gamma_2(v_f)$. This leads to a simple distributed version of the greedy algorithm: At the outset of each step, every facility computes its cost efficiency (i.e., the cost efficiency of its most cost efficient star). Facility v_f joins the dominating set if it has the best cost efficiency of all facilities in $\Gamma_2(v_f)$. Note that every such step can be implemented in 2 communication rounds using a local flooding.

The approximation ratio of the above distributed algorithm is identical to the sequential algorithm and thus asymptotically optimal. On the negative side, however, the algorithm's time complexity can be polynomial in the number of nodes. In fact, there are graphs in which in each step, only one facility is opened and some clients have to wait for a long time before being assigned. As pointed out in [134], this holds even in the MVC or MDS case, for example on the graph in Figure 6.2. An optimal vertex cover or dominating set consists of all nodes on the center axis. In the example graph, the *distributed greedy algorithm* manages to find this optimal vertex cover or dominating set, but selects nodes one-by-one from left to right. Hence, the running time of the algorithm can be as bad as $\Omega(\sqrt{n})$.

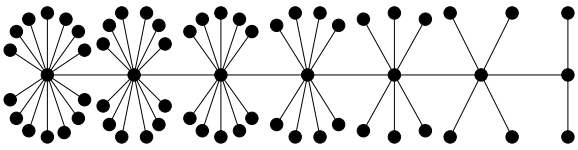

Figure 6.2: Simple distributed greedy algorithm: The *caterpillar graph* is a bad example for MVC and MDS.

One elegant way to solve the problem of avoiding long cascading waiting chains is to use randomization in each greedy-step [134]. In contrast, we

consider the approach of *fractionally* opening equally (or in fact, similarly) qualified facilities and fractionally connecting the corresponding clients. In this sense, our algorithm generalizes the distributed MDS algorithm proposed by Kuhn and Wattenhofer in [151] to the facility location problem.

6.3.1 Algorithm

At the heart of the algorithm is the distributed primal-dual technique which deterministically approximates LP_{FL} within a constant number of communication rounds. Each facility and client executes Algorithm 6.3 and 6.4, respectively, during which they keep track of the value of their primal variables x_{ij} and y_i. The algorithms consist of two nested loops which are both executed $h = \lceil\sqrt{k}\rceil$ times. The number of communication rounds in each iteration of the inner loop is constant, yielding the claimed constant time complexity of $O(k)$. Initially, all primal and dual variables y_i, x_{ij}, α_j, and β_{ij} are set to zero. Hence, the initial primal solution is infeasible, and the dual solution is feasible, yet far from optimal. During the course of the algorithm, both the primal and dual variables are gradually increased, thereby decreasing the primal infeasibility, and increasing dual optimality. At the end of the h^{th} iteration of the outer loop, the primal variables y_i and x_{ij} form a feasible solution to LP_{FL}.

A client v_j is called *uncovered* if it is not yet (fractionally) connected to a neighboring facility, i.e., $\sum_{v_i \in F(v_j)} x_{ij} < 1$. At any moment throughout the algorithm, the set of uncovered clients is called the *uncovered set* $\mathcal{A}$, initially $\mathcal{A} = C$. Further, we write $\mathcal{A}(v_i) := \mathcal{A} \cap F(v_i)$. Whenever a client v_j becomes covered, it sends a message $\mathcal{M}_j$ to facilities in $F(v_j)$. That way, the facilities always have a consistent view of the current $\mathcal{A}(v_i)$ (Lines 7 and 8 of the algorithm).

A *star* consists of one facility $v_i \in F$ and several uncovered, neighboring clients $v_j \in \mathcal{A}(v_i)$. The cost efficiency of a star B is the sum of the connection costs of the clients to facility v_i plus the facility cost f_i divided by the number of clients $|B|$. The *cost efficiency* $c(v_i)$ of a facility v_i is defined as the minimum star spanned from v_i, i.e.,

$$c(v_i) := \min_{B \in 2^{\mathcal{A}(v_i)}} \frac{f_i + \sum_{v_j \in B} c_{ij}}{|B|}. \tag{6.2}$$

The basic idea of the outer loop (s-loop) is to increase the y_i value of facilities with comparatively good cost efficiency $c(v_i)$[1]. More precisely, we call a facility *active* in a given iteration if its cost efficiency is at most $c(v_i) \leq \rho^{s/h}$. Only active facilities, that is, only facilities with good cost efficiency will execute the code between Lines 10 and 19. Particularly, only active facilities will increase their y_i value during an iteration. The idea of increasing the y_i value of facilities with good cost efficiency is inspired by the centralized

[1] In spite of there being exponentially many sets $B \in 2^{\mathcal{A}(v_i)}$, the facility can compute its cost efficiency $c(v_i)$ in polynomial time in Line 9 of Algorithm 6.3 by considering the clients ordered according to their connection costs c_{ij}.

```
1: h := ⌈√k⌉;
2: receive c_ij from all v_j ∈ C(v_i);
3: y_i := 0; A(v_i) := C(v_i);
4: for s := 1 to h by 1 do
5:    π_i^s := 0;
6:    for t := h − 1 to 0 by −1 do
7:       receive M_j from all v_j ∈ C(v_i);
8:       A(v_i) := A(v_i) \ {v_j ∈ C(v_i) | M_j = 1}
9:       c(v_i) := min_{B∈2^{A(v_i)}\{}} (f_i + Σ_{v_j∈B} c_ij)/|B|;
10:      if c(v_i) ≤ ρ^{s/h} then
11:         T_i := {v_j ∈ A(v_i) | c_ij ≤ ρ^{s/h}}
12:         (* Γ_i := (f_i + Σ_{v_j∈T_i} c_ij)/|T_i| *)
13:         if t = h − 1 then
14:            T_i^s := T_i;  Γ_i^s := Γ_i;  π_i^s := 1;
15:         end if
16:         Δy_i := max{y_i, m^{−t/h}} − y_i;
17:         y_i := y_i + Δy_i;
18:         send (y_i, f_i/|T_i|) to all v_j ∈ T_i
19:      end if
20:   end for
21:   forall v_j ∈ T_i^s do
22:      (* Δβ_ij := { 0          , ρ^{s/h} < c_ij
                     { ρ^{s/h} − c_ij , ρ^{s/h} ≥ c_ij ∧ π_i^s = 0
                     { Γ_i^s − c_ij   , ρ^{s/h} ≥ c_ij ∧ π_i^s = 1   *)
23:      (* β_ij := β_ij + Δβ_ij *)
24: end for
```

Algorithm 6.3: Facility v_i

greedy algorithm [125] that iteratively picks the facility with the best cost efficiency. In order to come up with fast, constant time algorithms in a distributed setting, this "greedy step" has to be parallelized. However, the greedy step's parallelization must be carefully implemented in order to avoid opening too many facilities at once, thus overly deteriorating the algorithm's performance.

We call a client v_j *tight* to an active facility v_i in iteration s of the outer loop if $c_{ij} \leq \rho^{s/h}$. That is, the *tight set* T_i in Line 11 consists of all clients that are connected to v_i by a connection of cost at most $\rho^{s/h}$. The significance of the tight set is that the increase of y_i in a given iteration results in an increase of x_{ij} of all clients v_j being in the tight set T_i. Since a client v_j may concurrently be in the tight set of several facilities, the increase of the different y_i must be handled with care. This is the role of the inner loop (t-loop), during which the y_i are gradually increased (Line 16) as long as the facility remains active.

The parameter ρ is a global parameter that depends on the given facility

```
1: h := ⌈√k⌉;
2: send c_ij to all v_i ∈ F(v_j);
3: α_j := 0;
4: ∀v_i ∈ F(v_j) : x_ij := 0;
5: for s := 1 to h by 1 do
6:    for t := h − 1 to 0 by −1 do
7:       M_j := 0;
8:       if Σ_{v_i∈F(v_j)} x_ij ≥ 1 then M_j := 1;
9:       send M_j to all v_i ∈ F(v_j);
10:      receive (y_i, f_i/|T_i|) from all v_i : v_j ∈ T_i;
11:      forall v_i : v_j ∈ T_i do
12:         if Σ_{v_i∈F(v_j)} x_ij < 1 then
13:            Δx_ij := y_i − x_ij;  x_ij := x_ij + Δx_ij;
14:         end for
15:      (* Δα_j := Σ_{v_i:v_j∈T_i} ( Δy_i f_i / |T_i| + Δx_ij c_ij ) ;  α_j := α_j + Δα_j; *)
16:   end for
17: end for
```

Algorithm 6.4: Client v_j

location instance, i.e., on the coefficients of the underlying linear program. The importance of ρ in the algorithm is to place an upper bound on the cost efficiency of a facility. More formally, ρ is defined as

$$\rho := \max_{v_j \in C} \min_{v_i \in F} (c_{ij} + f_i).$$

In a complete bipartite graph, the global parameters ρ, m, and n could be computed in 2 communication rounds by every node. In a general graph, however, computing these global parameters cannot be done locally. It is possible to circumvent this dependency on global knowledge by using a powerful technique introduced by Kuhn in Chapter 2 of [147]. Specifically, it is shown in [147] how nodes can use locally computable approximations $\hat{\rho}$ ($\hat{m}$, $\hat{n}$) to the globally exact values ρ (m, n), without deteriorating the algorithm's asymptotic time complexity and approximation guarantee. For ease of presentation, we present the algorithm assuming that the global constants ρ, n and m are known to every node. Also, we assume that $c_{ij} \geq 1$ and $f_i \geq 1$ for all $v_i \in F, v_j \in C$ in Section 6.3.2. The results are generalized to more general costs in Section 6.3.3.

6.3.2 Analysis

In a sense, our algorithm's analysis is based on the method of *dual fitting* [130] applied in a distributed setting. The basic idea of this method applied to FL can be described as follows: Using the linear program relaxation (LP) for facility location and its dual (DLP), we interpret our combinatorial algorithm as an algorithm that iteratively makes primal and dual updates in a

distributed fashion. Unfortunately, these updates do generally not lead to a feasible dual solution. However, the idea is to show that the objective function of the primal fractional solution computed by the algorithm is bounded by that of the dual. That is, the primal solution is fully paid for by the dual. By the laws of LP duality, it then remains to divide all dual values by a suitably large factor α that renders the dual variables feasible. The shrunk dual objective function is then a lower bound on OPT, and α is the algorithm's approximation guarantee. That is, instead of relaxing complementary slackness conditions as done in other primal-dual algorithms (e.g., [107, 235]), the *feasibility* of the dual itself is relaxed.

For notational clarity, we denote the increase of Δy_i in a certain iteration of the s and t-loop by $\Delta y_i(s,t)$ throughout this section. $\Delta x_{ij}(s,t)$, $\Delta\alpha_i(s,t)$, and $\Delta\beta_{ij}(s)$ are defined analogously.

We begin the analysis with the observation that the resulting solution is feasible.

Lemma 6.7. *Algorithms 6.3 and 6.4 produces a feasible primal solution for LP_{FL}.*

Proof. The feasibility of the second LP condition, $y_i - x_{ij} \geq 0$, $\forall v_j \in C, i \in F$, directly follows from the definition of the algorithm. Specifically, in Line 13 of Algorithm 6.4, the value of a connection variable x_{ij} never exceeds the value of the corresponding y_i.

As for the first LP condition, observe that in Lines 12 and 13, the x_{ij} values of client v_j are set to the corresponding y_i as long as $\sum_{v_i \in F(v_j)} x_{ij} < 1$. Hence, as soon as the sum of the y_i values of facilities to which v_j is tight exceeds 1, the sum of the x_{ij} variables of v_j also exceeds 1. Formally, we have that $\sum_{v_i : v_j \in T_i} y_i \geq 1$ implies $\sum_{v_i \in F(v_j)} x_{ij} \geq 1$.

Now, assume for contradiction that v_j is a client which is still uncovered at the end of the algorithm, i.e., $\sum_{v_i : v_j \in T_i} y_i < 1$. Consider the very last iteration of the inner loop ($s = h$, $t = 0$). By the definition of ρ, there exists at least one facility $v_i \in F(v_j)$ with cost efficiency $c(v_i) \leq \rho$ covering client v_j. Because $s = h$, facility v_i becomes *active* and increases its y_i value to $m^{-t/h} = 1$ in Lines 16 and 17. Subsequently, v_j sets $x_{ij} := 1$ which contradicts the assumption that $v_j \in \mathcal{A}(v_i)$ at the end of the algorithm. □

If a facility is active in a certain iteration, its cost-efficiency $c(v_i)$ is, by definition, at most $\rho^{s/h}$. The tight set T_i does not necessarily contain the same clients which constituted the optimal cost-efficiency. Although the cost efficiency of T_i may be larger than $c(v_i)$, $\Gamma_i \geq c(v_i)$, the next lemma shows that the cost-efficiency of the tight set T_i, Γ_i, is at most $\rho^{s/h}$.

Lemma 6.8. *In every iteration of the t-loop, if $c(v_i) \leq \rho^{s/h}$ for a facility v_i, then $\Gamma_i \leq \rho^{s/h}$.*

Proof. Let the set B be the optimal star that constituted $c(v_i)$. First, observe that if $c(v_i) \leq \rho^{s/h}$, and because B minimizes $c(v_i)$, no client $v_j \in B$ can

have connection cost $c_{ij} > \rho^{s/h}$. Let $Q := T_i \setminus B$ be the set of clients $v_j \notin B$ with $c_{ij} \leq \rho^{s/h}$. Γ_i is upper bounded by

$$\begin{aligned} \Gamma_i &= \frac{f_i + \sum_{v_j \in T_i} c_{ij}}{|T_i|} = \frac{f_i + \sum_{v_j \in B} c_{ij} + \sum_{v_j \in Q} c_{ij}}{|T_i|} \\ &\underset{c(v_i) \leq \rho^{s/h}}{\leq} \frac{|B|\rho^{s/h} + |Q|\rho^{s/h}}{|T_i|} = \rho^{s/h}. \end{aligned}$$

□

Bounding the primal objective function by the dual objective function is key to applying the method of dual fitting. The next lemma provides such a bound by showing that throughout the execution of the algorithm, the values of the primal and dual objective functions are equal.

Lemma 6.9. *At the end of each t-loop, it holds that*

$$\sum_{v_j \in C} \alpha_j = \sum_{v_j \in C, v_i \in F} c_{ij} x_{ij} + \sum_{v_i \in F} f_i y_i. \tag{6.3}$$

Proof. We prove the claim by induction over the iterations of the t-loop. At the beginning of the algorithms, both sides of the equation are 0. Assume that the claim is true before starting a new iteration s'. If facility v_i increases its y_i during s', tight clients $v_j \in T_i$ may increase their corresponding x_{ij} as well. Hence, the right hand side of (6.3) increases by

$$\Delta RHS = \sum_{i \in F} \Delta y_i f_i + \sum_{j \in C, i \in F} \Delta x_{ij} c_{ij}.$$

As for the left hand side of (6.3), Line 15 of Algorithm 6.4 defines the increase of the clients α_j values. Its sum $\sum_{j \in C} \alpha_j$ therefore increases as

$$\begin{aligned} \Delta LHS &= \sum_{v_j \in C} \sum_{v_i : v_j \in T_i} \left(\frac{\Delta y_i f_i}{|T_i|} + \Delta x_{ij} c_{ij} \right) \\ &= \sum_{v_i \in F} \sum_{v_j \in T_i} \frac{\Delta y_i f_i}{|T_i|} + \sum_{v_j \in C} \sum_{v_i \in F} \Delta x_{ij} c_{ij} \\ &= \sum_{v_i \in F} \Delta y_i f_i + \sum_{v_j \in C, v_i \in F} \Delta x_{ij} c_{ij} = \Delta RHS. \end{aligned}$$

□

In the next lemma, we characterize the steady increase of a facility v_i's cost efficiency during the course of the algorithm.

Lemma 6.10. *At the beginning of each iteration of the s-loop, it holds for all facilities $v_i \in F$ that $c(v_i) \geq 1$ for $s = 1$ and $c(v_i) > \rho^{s-1/h}$ for $s > 1$.*

Proof. The case $s = 1$ follows from the assumption that $c_{ij} \geq 1$ and $f_i \geq 1$ (cf. Section 6.3.3). Consider iteration $s > 1$ and let $s' = s - 1$. For all facilities with $c(v_i) > \rho^{s'/h}$, the claim holds. In the last t-loop iteration of the s'^{th} iteration, all facilities v_i with $c(v_i) \leq \rho^{s'/h}$ set y_i to $m^{-0/k} = 1$. Consequently, all v_j in the tight set T_i of iteration s' become covered and leave $\mathcal{A}$. It follows that for such a facility v_i, it holds that $\forall v_j \in \mathcal{A}(v_i) : \; c_{ij} > \rho^{s'/h}$. The claim now follows from

$$c(v_i) \;=\; \min_{B \in 2^{\mathcal{A}(v_i)} \setminus \{\}} \frac{f_i + \sum_{v_j \in B} c_{ij}}{|B|} \;>\; \frac{f_i}{|B|} + \frac{|B|\rho^{s-1/h}}{|B|} \;>\; \rho^{s-1/h}.$$

□

A client may be tight to several facilities. If all these facilities simultaneously increased their y_i values, the dual value α_j of v_j may increase too much. Consider an iteration of the s-loop. During the early iterations of the t-loop, the increase in Δy_i of active facilities is small, because t is close to h. Intuitively, it is acceptable if a client is tight to many active facilities in these early iterations. In other words, the higher the increases Δy_i of active facilities, the fewer active facilities a client is allowed to be tight to. The following lemma establishes precisely this relationship.

Lemma 6.11. *Let $A_j := \{v_i \mid v_j \in T_i\}$ be the active set for an uncovered client v_j. At the beginning of each iteration of the t-loop,*

$$|A_j| \;\leq\; m^{t+1/h}.$$

Proof. From the previous iteration of the loop, we know that for each active facility $v_i \in A_j$, it holds that $y_i \geq m^{-(t+1)/h}$. Now, assume for contradiction that $|A_j| > m^{t+1/h}$ for some $v_j \in C$. If so, then

$$\sum_{v_i \in A_j} y_i \;\geq\; |A_j| \cdot m^{-(t+1)/h} \;>\; 1$$

and consequently $\sum_{v_i \in F(v_j)} x_{ij} > 1$ in Line 13 of Algorithm 6.4 of the same loop iteration. This contradicts the assumption that client v_j is uncovered and thus, the claim follows. □

In the next lemma, we bound the total amount of $\Delta\alpha_j$ that each client can receive in one iteration of the s-loop. For that purpose, let $\Delta\alpha_j(s) := \sum_{t=0}^{h-1} \Delta\alpha_j(s,t)$ be the increase of α_j during the s^{th} iteration of the outer loop. Let $T_i(s,t)$ be the set of clients that are tight to v_i in the iterations s and t. Further, we define

$$\sigma_j(s) := \sum_{t=1}^{h} \sum_{v_i \in A_j(s,t)} \Delta y_i.$$

Intuitively, $\sigma_j(s)$ is the increase of the y_i value at facilities to which client v_j has been tight during the course of the s^{th} iteration of the outer loop. The following lemma relates $\alpha_j(s)$ and $\sigma_j(s)$.

Lemma 6.12. *The sum of the $\Delta\alpha_j$ values collected in iteration s at a node v_j is upper bounded by*

$$\Delta\alpha_j(s) \;\leq\; (\sigma_j(s)+2)\cdot\rho^{s/h}.$$

Proof. Applying Lemma 6.8 and by the definition of $\Delta\alpha_j$, we have

$$\begin{aligned}
\Delta\alpha_j(s) &\leq \sum_{t=0}^{h-1}\Delta\alpha_j(s,t)\\
&\leq \sum_{t=0}^{h-1}\sum_{v_i\in A_j(s,t)}\frac{\Delta y_i f_i}{|T_i|}+\sum_{t=0}^{h-1}\sum_{v_i\in A_j(s,t)}\Delta x_{ij}c_{ij}\\
&\leq \sum_{t=0}^{h-1}\sum_{v_i\in A_j(s,t)}\Delta y_i\Gamma_i+\sum_{t=0}^{h-1}\sum_{v_i\in A_j(s,t)}\Delta x_{ij}c_{ij}.
\end{aligned}$$

The connection cost c_{ij} between a client v_j and any facility $v_i\in A_j(s,t)$ is by definition at most $\rho^{s/h}$. Moreover, by Lemma 6.8, we know that $\Gamma_i\leq\rho^{s/h}$ in every iteration of the s-loop. When plugging in the definition of $\sigma_j(s)$, we therefore have

$$\Delta\alpha_j(s) \;\leq\; \rho^{s/h}\left(\sigma_j(s)+\sum_{t=0}^{h-1}\sum_{v_i\in A_j(s,t)}\Delta x_{ij}\right).$$

Finally, we need to bound the term $\sum_{t=0}^{h-1}\sum_{i\in A_j(s,t)}\Delta x_{ij}$. By Lines 12 and 13 of the client algorithm, the values x_{ij} are increased only as long as $\sum_{v_i\in A_j(s,t)}x_{ij}<1$ and therefore, the sum of all but the last non-zero Δx_{ij} values cannot exceed 1. Because the final non-zero Δx_{ij} is clearly at most 1, it follows that $\sum_{t=0}^{h-1}\sum_{v_i\in A_j(s,t)}\Delta x_{ij}\leq 2$, which concludes the proof. □

Next, we want to find bounds for $\sigma_j(s)$. Assume that client v_j becomes covered during iteration s_j^* of the outer loop. Notice that for every client v_j, there is exactly one iteration s_j^*. Once covered, v_j will not be in $\mathcal{A}$ and therefore, not in any T_i. Consequently, $\sigma_j(s')=0$ for all $s'>s_j^*$. The other two cases, $s'<s_j^*$ and $s'=s_j^*$ are subject of the following lemma.

Lemma 6.13. *For all iterations of the s-loop, it holds that*

$$\begin{aligned}
\sigma_j(s) &\leq 1 && \forall s\neq s_j^*\\
\sigma_j(s) &\leq m^{1/h} && s=s_j^*
\end{aligned}$$

Proof. The first case, $s' < s_j^*$, follows from the definition of $\sigma_j(s)$. If

$$\sigma_j(s') \;=\; \sum_{t=1}^{h} \sum_{v_i \in A_j(s,t)} \Delta y_i \;\geq\; 1,$$

then v_j would have become covered in iteration s' and hence, $s' = s_j^*$.

It remains to analyze the iteration during which v_j becomes covered, i.e., $s' = s_j^*$. Consider the iterations of the inner loop during iteration s_j^* of the outer loop. Let t^* denote the iteration during which v_j becomes covered. Clearly, for all $t' > t^*$, it holds that $\sum_{v_i : v_j \in T_i(t')} \Delta y_i(t') = 0$ because v_j is already covered. Hence, we only need to analyze the first t^* iterations of the inner loop. Summing up all increases, we get

$$\begin{aligned}
\sigma_j(s') &= \sum_{t=t^*+1}^{h-1} \sum_{v_i \in A_j(t)} \Delta y_i(t) + \sum_{v_i \in A_j(t^*)} \Delta y_i(t^*) \\
&\leq 1 + \sum_{v_i \in A_j(t^*)} \Delta y_i(t^*) \\
&\leq 1 + \left(m^{-t/h} - m^{-(t+1)/h}\right) \cdot |A_j(t^*)| \\
&\leq 1 + \left(m^{-t/h} - m^{-(t+1)/h}\right) \cdot m^{t+1/h} \\
&\leq 1 + \left(m^{1/h} - 1\right) = m^{1/h}.
\end{aligned}$$

The first inequality follows from the fact that by definition of t^*, client v_j is not covered after the iterations $h-1, \ldots, t^*+1$. The second inequality holds because a facility cannot be active in an iteration t^* if it has not been active in iteration t^*-1 already. Therefore, the increase of y_i of active facilities v_i is at most $m^{-t/h} - m^{-(t+1)/h}$. The third inequality follows from Lemma 6.11. □

For the dual solution to be feasible, the linear program condition imposes that $\sum_{v_j \in C(v_i)} \beta_{ij} \leq f_i$ holds for every facility $v_i \in F$. Unfortunately, the dual solution produced by our algorithm does not exhibit this feasibility property. However, we can at least show that the degree of infeasibility is bounded. Specifically, it holds that the sum of the increases of the β_{ij} in *a single iteration* of the s-loop, it fulfils the desired property.

Lemma 6.14. *For all $v_i \in F$ and all iterations s of the outer loop, it holds that*

$$\sum_{v_j \in C(v_i)} \Delta\beta_{ij}(s) \;\leq\; f_i.$$

Proof. We distinguish two cases, depending on whether $\pi_i(s)$ equals 0 or 1. In the first case, $\pi_i(s) = 0$, the facility v_i's cost efficiency was insufficient to

increase its y_i value during the s^{th} iteration. We therefore have

$$\sum_{v_j \in C(v_i)} \Delta\beta_{ij}(s) \quad = \quad \sum_{v_j \in T_i} (\rho^{s/h} - c_{ij}) \;=\; \rho^{s/h}|T_i| - \sum_{v_j \in T_i} c_{ij}$$

Assume for contradiction that $\sum_{v_j \in C(v_i)} \Delta\beta_{ij}(s) > f_i$ for some facility v_i and iteration s. It follows that

$$\rho^{s/h} \;>\; \frac{f_i + \sum_{v_j \in T_i} c_{ij}}{|T_i|} \;\geq\; c(v_i),$$

which in turn implies $\pi_i(s) = 1$ for $B = T_i$. This establishes the contradiction.

As for the second case, $\pi_i(s) = 1$, we have

$$\sum_{v_j \in C(v_i)} \Delta\beta_{ij}(s) = \sum_{v_j \in T_i} (\Gamma_i^s - c_{ij}) = |T_i| \cdot \frac{f_i + \sum_{v_j \in T_i} c_{ij}}{|T_i|} - \sum_{v_j \in T_i} c_{ij} = f_i.$$

Therefore, the lemma holds in both cases. □

Having bounded the degree of infeasibility of one dual constraint of LP_{FL}, it now remains to do the same for the other one, $\alpha_j - \beta_{ji} \leq c_{ij}$, for all $v_j \in C$ and $v_i \in F(v_j)$. Again, we give weaker bounds that do not hold for the entire execution of the algorithm, but merely for a single iteration of the outer loop.

Lemma 6.15. *Let $\Delta\alpha_j(s)$ be the sum of the $\Delta\alpha_j(t)$ over all iterations of the t-loop in the s^{th} iteration of the s-loop. For all $v_j \in C$, $v_i \in F(v_j)$, and all iterations s, it holds that*

$$\frac{\Delta\alpha_j(s)}{(m^{1/h}+2)\rho^{1/h}} - \Delta\beta_{ij}(s) \quad \leq \quad c_{ij}$$

Proof. We distinguish three cases, depending on how much increase of β_{ij} was assigned to the connection between v_i and v_j in Line 22 of the facility algorithm. Regardless of the specific case, the value $\Delta\alpha_j(s)$ is bounded by $\Delta\alpha_j(s) \leq \rho^{s/h}(m^{1/h}+2)$ due to Lemmas 6.12 and 6.13.

1) In the case $\rho^{s/h} \leq c_{ij}$, the algorithm sets $\Delta\beta_{ij}(s)$ to 0. Therefore

$$\frac{\Delta\alpha_j(s)}{(m^{1/h}+2)\rho^{1/h}} - \Delta\beta_{ij}(s) \;\leq\; \rho^{(s-1)/h} \;\leq\; c_{ij}.$$

2) In the second case, the client j is tight to i, i.e., $\rho^{s/h} > c_{ij}$, but $\pi_i(s) = 0$. Plugging in the corresponding value for $\Delta\beta_{ij}(s)$, we get

$$\frac{\Delta\alpha_j(s)}{(m^{1/h}+2)\rho^{1/h}} - \left(\rho^{s/h} - c_{ij}\right) \quad \leq \quad \rho^{(s-1)/h} - \rho^{s/h} + c_{ij} \;\leq\; c_{ij}.$$

3) Finally, consider the last case, $\rho^{s/h} > c_{ij}$ and $\pi_i(s) = 1$. Substituting $\Delta\beta_{ij}(s)$ by $\Gamma_i^s - c_{ij}$ yields

$$\Delta\beta_{ij}(s) + c_{ij} \;=\; \Gamma_i^s - c_{ij} + c_{ij} \;=\; \Gamma_i^s \;\geq\; c(v_i).$$

For $s > 1$, we know by Lemma 6.10, that $c(v_i) \geq \rho^{(s-1)/h}$ at the beginning of iteration s, hence

$$\Delta\beta_{ij}(s) + c_{ij} \;\geq\; \rho^{(s-1)/h}.$$

Using the above inequality, we obtain

$$\frac{\Delta\alpha_j(s)}{(m^{1/h}+2)\rho^{1/h}} \quad\leq\quad \rho^{(s-1)/h} \;\leq\; \Delta\beta_{ij}(s) + c_{ij}.$$

Subtracting $\Delta\beta_{ij}(s)$ concludes the proof for the case $s > 1$. The case $s = 1$ follows similarly. By Lemma 6.10, we can lower bound $c(v_i) \geq 1$ and therefore $\Delta\beta_{ij}(s) + c_{ij} \geq 1$. The claim now follows from

$$\frac{\Delta\alpha_j(s)}{(m^{1/h}+2)\rho^{1/h}} \;\leq\; \rho^{(s-1)/h} \;=\; 1 \;\leq\; \Delta\beta_{ij}(s) + c_{ij}.$$

□

Having bounded the degree of dual infeasibility in the two previous lemmas, we can now establish the approximation ratio of the algorithm using the laws of LP duality. Specifically, the dual feasibility is violated only by a factor $O(h(m\rho)^{1/h})$ and hence, when dividing α_j and β_{ij} by suitably large values, we obtain a feasible solution $\hat{\alpha}_j$ and $\hat{\beta}_{ij}$.

Theorem 6.16. *For an arbitrary integer $k > 0$, the algorithm computes an $O(\sqrt{k}(m\rho)^{1/\sqrt{k}})$ approximation to the fractional facility location problem in $O(k)$ communication rounds.*

Proof. The running time follows from the definition of the algorithm. For the analysis of the approximation ratio, let $\hat{\alpha}_j$ and $\hat{\beta}_{ij}$ be defined as

$$\hat{\alpha}_j := \frac{\alpha_j}{h(m^{1/h}+2)\rho^{1/h}} \quad\text{and}\quad \hat{\beta}_{ij} := \frac{\beta_{ij}}{h},$$

respectively. We show that the variables $\hat{\alpha}_j$ and $\hat{\beta}_{ij}$ form a feasible solution to the dual LP. The feasibility of the second dual constraint follows from Lemma 6.14. Particularly, it holds that $\sum_{v_j \in C(v_i)} \beta_{ij}(s) \leq f_i$ for all iterations s and all facilities $v_i \in F$. As a consequence, we obtain $\sum_{v_j \in C(v_i)} \beta_{ij} \leq h \cdot f_i$ and therefore, $\sum_{v_j \in C(v_i)} \hat{\beta}_{ij} \leq f_i$.

Next, we show the feasibility of the first constraint by bounding $\hat{\alpha}_j - \hat{\beta}_{ij}$ as

$$\begin{aligned}\hat{\alpha}_j - \hat{\beta}_{ij} &= \frac{\sum_{s=0}^{h-1}\alpha_j(s)}{h(m^{1/h}+2)\rho^{1/h}} - \frac{\sum_{s=0}^{h-1}\beta_{ij}(s)}{h} \\ &= \frac{1}{h}\sum_{s=0}^{h-1}\left(\frac{\alpha_j(s)}{(m^{1/h}+2)\rho^{1/h}} - \beta_{ij}(s)\right).\end{aligned}$$

By Lemma 6.15, each term of the sum is bounded by c_{ij}. Therefore, we have $\hat{\alpha}_j - \hat{\beta}_{ij} \leq hc_{ij}/h \leq c_{ij}$.

Let OPT and ALG denote the optimal value and the value as computed by the algorithm, respectively. By LP duality, the sum of the $\hat{\alpha}_j$ values is a lower bound for OPT. As for ALG, recall that by Lemma 6.9, we know that the value of the primal and dual objective function is equal at the end of the algorithm. Therefore, we can bound ALG as

$$\begin{aligned} ALG &= \sum_{v_i \in F} f_i y_i + \sum_{v_i \in F} \sum_{v_j \in C} c_{ij} x_{ij} \\ &\underset{\text{Lemma 6.9}}{=} \sum_{v_j \in C} \alpha_j \;\leq\; h(m^{1/h} + 2)\rho^{1/h} \sum_{v_j \in C} \hat{\alpha}_j \\ &\leq h(m^{1/h} + 2)\rho^{1/h} \cdot OPT \;\in\; O(h(m\rho)^{1/h}) \cdot OPT. \end{aligned}$$

Finally, the theorem follows from $h = \lceil \sqrt{k} \rceil$. □

Remark It is possible to express the results of this section in a tighter form. In particular, the algorithm's approximation guarantee can equally be expressed in terms of Δ_m and Δ_n instead of m and n, respectively, where Δ_m and Δ_n denote the largest degree of any facility or client in the graph.

6.3.3 Arbitrary Coefficients

The algorithm and analysis of Sections 6.3.1 and 6.3.2 is based on the assumption that $c_{ij} \geq 1$ and $f_i \geq 1$ for all $v_j \in C, v_i \in F$. In this section, we show how to handle the general case in which connection and opening costs can be arbitrary non-negative values. Furthermore, this technique can be used to get rid of the dependency on ρ in the approximation ratio.

The idea is to initially scale all costs such that the above condition holds. The problem is that the straightforward approach of multiplying all costs with the minimum c_{ij} or f_i might overly blow up the coefficient ρ, or it may even be infeasible for zero valued costs. For that reason, we need to perform a more subtle scaling that is inspired by a similar technique given in [32].

The parameter ρ is a lower bound for the objective value OPT of the optimal solution. Because there are n clients, all stars with cost-efficiency smaller than ρ/n can be added to a solution ALG, incurring costs at most OPT. This observation motivates the following scaling procedure performed at the beginning of the algorithm.

1. For every facility v_i, choose the largest set B_i of clients such that $(f_i + \sum_{v_j \in B_i} c_{ij})/|B_i| \leq \rho/n$. If such a B_i exists, set $y_i := 1$ and $x_{ij} := 1$ for all $v_j \in B_i$. Let C' be the set of unconnected clients.

2. For all $v_j \in C'$ and $v_i \in F$, set $c'_{ij} := nc_{ij}/\rho$ and $f'_i := nf_i/\rho$, respectively. Clients in $C \setminus C'$ do not participate in the algorithm further.

3. Execute Algorithms 6.3 and 6.4 with clients and facilities in C' and F, the coefficient $\rho' = n$ (the new ρ), and costs c'_{ij} and f'_i.

Notice that—if we assume that every node knows n—the above procedure can be executed in our distributed model in a constant number of communication rounds. The facility location instance resulting from the above transformation fulfils the following useful and simple property.

Lemma 6.17. *Consider a facility location instance derived from the above transformation. Throughout the algorithm and for all $v_i \in F$, it holds that $c(v_i) \geq 1$.*

Proof. If B is the set of clients constituting $c(v_i)$, then

$$c(v_i) \;=\; \frac{f'_i + \sum_{v_j \in B} c'_{ij}}{|B|} \;=\; \frac{\frac{n}{\rho}\left(f_i + \sum_{v_j \in B} c_{ij}\right)}{|B|}$$

If we assume for contradiction that $v_i \in F$ and $c(v_i) < 1$, it follows that $\frac{f_i + \sum_{v_j \in B} c_{ij}}{|B|} < \frac{\rho}{n}$. This contradicts the fact that the star B was *not* selected during the transformation, that is, the clients $v_j \in B$ are in C'. □

Having Lemma 6.17 allows us to apply Lemmas 6.10 and 6.15 as in the proof of Section 6.3.2. The remainder of the proof in Section 6.3.2 remains the same.

In summary, the transformed algorithm runs in $O(k)$ communication rounds and yields a solution of cost at most $O(\sqrt{k}(mn)^{1/\sqrt{k}}) \cdot OPT + OPT$ for the fractional facility location problem LP_{FL}.

6.4 Distributed Randomized Rounding

The algorithms presented in Sections 6.2 and 6.3 merely compute approximations to the *fractional* facility location problem. In order to come up with a solution to the *integer* facility location problem, the fractional solutions obtained in the previous section need to be rounded. Clearly, the randomized rounding step should neither overly increase the total opening costs, nor the total connection costs. Interestingly, randomized rounding schemes that have been well-studied in the sequential case can also be implemented in distributed settings. The idea for the randomized rounding is inspired by the filtering technique introduced in [165].

In the following, let x_{ij} and y_i be the values obtained from a fractional facility location algorithm. The variables $\hat{x}_{ij}$ and $\hat{y}_i$ denote the corresponding rounded integer values. For every client $v_j \in C$, let $C_j^* := \sum_{v_i \in F(v_j)} c_{ij} x_{ij}$ be the weighted cost of v_j's connections. Further, let the *log-neighborhood* $\Lambda(v_j)$ of a client v_j be the set of all facilities that are located within a factor of $\log(n+m)$ of the weighted connection cost. Formally, for every $v_j \in C$, $\Lambda(v_j) := \{v_i \in F(v_j) \mid c_{ij} \leq \log(n+m) \cdot C_j^*\}$. The idea is to round the

INPUT: fractional solution y_i from Algorithm 6.3
OUTPUT: integral solution $\hat{y}_i$ to ILP
1: $p_i := \min\{1, y_i \cdot \ln(n+m)\}$;
2: $\hat{y}_i := \begin{cases} 1 & \text{, with probability } p_i \\ 0 & \text{, with probability } 1-p_i \end{cases}$
3: **send** $\hat{y}_i$ to all clients $v_j \in C(v_i)$;
4: **if receive** JOIN-MSG **then** $\hat{y}_i := 1$;

Algorithm 6.5: Randomized Rounding - Facility

fractional values y_i at each facility v_i in such a way that with high probability, all clients have at least one open facility in their log-neighborhood. The set of open facilities in a client's log-neighborhood is denoted by $Open(v_j)$. Each client then simply connects itself to the open facility in $\Lambda(v_j)$ with minimum connection cost c_{ij}. In case some v_j has no open facility in $\Lambda(v_j)$, it asks the "cheapest" facility to open itself. Algorithms 6.5 and 6.6 present the scheme in more detail.

Theorem 6.18. *Let x_{ij} and y_i for each $v_j \in C$ and $v_i \in F$ be a feasible solution for the fractional facility location problem LP_{FL} with cost at most $\alpha \cdot OPT$. In two rounds of communication, Algorithms 6.5 and 6.6 produce a feasible integer solution $\hat{x}_{ij}$, $\hat{y}_i$ to ILP_{FL} with expected cost $O(\log(m+n))\alpha \cdot OPT$.*

Proof. It follows from the definition of C_j^* and $\Lambda(v_j)$ that

$$\sum_{v_j \in F \setminus \Lambda(v_j)} x_{ij} \;\leq\; \frac{1}{\log(n+m)},$$

because otherwise, C_j^* would be larger. Any feasible solution to LP_{FL} must maintain the invariants $\sum_{v_i \in F(v_j)} \hat{x}_{ij} \geq 1$ and $\sum_{v_i \in \Lambda(v_j)} \hat{x}_{ij} \leq \sum_{v_i \in \Lambda(v_j)} \hat{y}_i$. Therefore,

$$\sum_{v_i \in \Lambda(v_j)} \hat{y}_i \;\geq\; \sum_{v_i \in \Lambda(v_j)} \hat{x}_{ij} \;\geq\; 1 - \frac{1}{\log(n+m)}. \tag{6.4}$$

For each client v_j having $Open(v_j) \neq \emptyset$, the connection costs are at most $c_{ij} \leq \ln(n+m) C_j^*$ by the definition of the neighborhood $\Lambda(v_j)$. These clients thus account for total connection costs of at most $\ln(n+m) \sum_{v_j \in C, v_i \in F} c_{ij} x_{ij}$. A facility declares itself open in Line 2 of Algorithm 6.5 with a probability of $\min\{1, y_i \cdot \ln(n+m)\}$. The expected opening costs of facilities opened in Line 2 are thus bounded by $\ln(n+m) \sum_{v_i \in F} y_i f_i$.

It remains to bound the costs entailed by clients that are *not* covered, i.e. $Open(v_j) = \emptyset$, and facilities that are opened via a JOIN-MSG message. The probability q_j that a client v_j does *not* have an open facility in its log-

INPUT: fractional solution x_{ij} from Algorithm 6.4
OUTPUT: integral solution $\hat{x}_{ij}$ to ILP

1: $C_j^* := \sum_{v_i \in F(v_j)} c_{ij} x_{ij}$;
2: $\Lambda(v_j) := \{v_i \in F(v_j) \mid c_{ij} \leq \ln(n+m) \cdot C_j^*\}$;
3: **receive** y_i from all $v_i \in F(v_j)$;
4: $Open(v_j) := \Lambda(v_j) \cap \{v_i \in F(v_j) \mid y_i = 1\}$;
5: **if** $Open(v_j) \neq \emptyset$ **then**
6: $\quad v_i' := \text{argmin}_{v_i \in \Lambda(v_j)} c_{ij}$; $\hat{x}_{i'j} := 1$;
7: **else**
8: $\quad v_i' := \text{argmin}_{v_i \in F} (c_{ij} + f_i)$;
9: $\quad$ **send** JOIN-MSG to facility v_i'; $\hat{x}_{i'j} := 1$;
10: **fi**

Algorithm 6.6: Randomized Rounding - Client

neighborhood is at most

$$
\begin{aligned}
q_j \quad &= \quad \prod_{v_i \in \Lambda(v_j)} (1 - p_i) \;=\; \left(\sqrt[n+m]{\prod_{v_i \in \Lambda(v_j)} (1 - p_i)} \right)^{n+m} \\
&\leq \quad \left(\frac{\sum_{v_i \in \Lambda(v_j)} (1 - p_i)}{n+m} \right)^{n+m} \\
&\underset{|\Lambda(v_j)| \leq m}{\leq} \quad \left(1 - \frac{\ln(n+m) \sum_{v_i \in \Lambda(v_j)} y_i}{n+m} \right)^{n+m} \\
&\underset{\text{Eq. (6.4)}}{\leq} \quad \left(1 - \frac{\ln(n+m)}{n+m} \left(1 - \frac{1}{\ln(n+m)} \right) \right)^{n+m} \\
&= \quad \left(1 - \frac{\ln(n+m) - 1}{n+m} \right)^{n+m} \\
&\leq \quad e^{-\ln(n+m)-1} \;\leq\; \frac{1}{e(n+m)}. \qquad (6.5)
\end{aligned}
$$

The first inequality follows from the fact that for every sequence of positive numbers, the geometric mean is smaller than or equal to the arithmetic mean of these numbers.

An uncovered client v_j sends a JOIN-MSG message to the facility $v_i \in F(v_j)$ that minimizes $c_{ij} + f_i$. Each of these costs is at most $\sum_{v_j \in C, v_i \in F} c_{ij} x_{ij} + \sum_{v_i \in F} y_i f_i$ because $\underline{x}$ and $\underline{y}$ would not constitute a feasible solution other-

wise. Combining this with the above results, the total expected cost is

$$\begin{aligned} E[ALG] &\leq \ln(n+m)\left(\sum_{v_j\in C, v_i\in F} c_{ij}x_{ij} + \sum_{v_i\in F} y_i f_i\right) + \\ &\quad + \frac{n}{e(n+m)}\left(\sum_{v_j\in C, v_i\in F} c_{ij}x_{ij} + \sum_{v_i\in F} y_i f_i\right) \\ &\leq (\ln(n+m) + O(1))\,\alpha\cdot OPT, \end{aligned}$$

which concludes the proof of Theorem 6.18. □

When using this distributed randomized rounding scheme with the approximation algorithms for the fractional FL problem presented in Sections 6.2 and 6.3, we obtain local approximation algorithms for ILP_{FL}. Their performance is summarized in the following corollaries.

Corollary 6.19. *In the $\mathcal{LOCAL}$ model, Algorithm 6.2 in combination with randomized rounding achieves an approximation ratio of $O(m^{1/k}\log(n+m))$ to ILP_{FL} in $O(k)$ communication rounds.*

Corollary 6.20. *In the $\mathcal{CONGEST}$ model, Algorithms 6.3 and 6.4 in combination with randomized rounding achieve an approximation ratio that is in the order of $O(\sqrt{k}(m\rho)^{1/\sqrt{k}}\cdot\log(n+m))$ to ILP_{FL} in $O(k)$ communication rounds, where ρ depends on the coefficients of the problem instance.*

Corollary 6.19 implies that in order to achieve an $O(m^{1/k}\log(n+m))$ approximation to the distributed non-metric facility location problem, nodes only need to have knowledge about their $O(k)$-hop neighborhood. Furthermore, an $O(\log(m+n))$ approximation to ILP_{FL} can be computed in $O(\log m)$ time in the $\mathcal{LOCAL}$ model, whereas our algorithm in the $\mathcal{CONGEST}$ model achieves only an approximation ratio of $O(\log(m\rho)\log(n+m))$ in time $O(\log^2(m\rho))$.

Chapter 7

Local Computation: Lower Bounds

Studying lower bounds and impossibility results has a long tradition in the theory of distributed computing [90]. When it comes to distributed approximation or the distributed complexity of problems in the message passing model in general, however, interesting lower bounds become rarer. This is particularly true for the unrestricted $\mathcal{LOCAL}$ model. In fact, most of the well-known lower bounds in this area of distributed computing are based on restrictions on message reception in the radio network model (e.g. [10, 155]) or on bandwidth restrictions in the $\mathcal{CONGEST}$ model. In particular, the lower bounds on the distributed computation (or approximation) of the MST fall into the latter category [76, 197]. The only general lower bound on local computation derived solely on the limitations of *locality* is the pioneering and seminal lower bound by Linial [166], which showed that the non-uniform $O(\log^* n)$ coloring algorithm by Cole and Vishkin [58] is asymptotically optimal for the ring. This lower bound has been cherished by researchers as a fundamental advancement in the theory of distributed computation.

Linial's lower bound is based on the *drosophila melanogaster* of distributed computing, the ring network. For simple optimization problems such as minimum vertex cover, however, highly symmetric graphs such as rings often feature a straight-forward solution with constant approximation ratio. In any δ-regular graph, for example, the algorithm which includes all nodes in the vertex cover is already a 2-approximation for MVC: Each node will cover at most δ edges, the graph has $n\delta/2$ edges, and therefore at least $n/2$ nodes need to be in the minimum vertex cover. On the other extreme, asymmetric graphs often enjoy constant-time algorithms, too. In a tree, choosing all inner nodes yields a 2-approximation. The same trade-off exists for node degrees. If the maximum node degree is low (constant), we can tolerate to choose all nodes, and have by definition a good (constant) approximation. If there are nodes with high degree, the diameter of the graph is small, and

a few communication rounds suffice to inform all nodes of the entire graph. So, what could be a hard instance for an optimization problem like MVC? If there exists such an instance at all, it needs to be based on the construction of a not too symmetric and not too asymmetric graph with a variety of node degrees! Not many graphs with these "non-properties" are known in distributed computing.

In this chapter, we construct such a graph in order to derive the lower bound on the achievable time-approximation trade-off by any distributed algorithm. In particular, we show that if every node has information about its k-hop neighborhood (or alternatively, if nodes employ a distributed algorithm with running time at most k), MVC cannot be approximated better than by a multiplicative factor of $\Omega(n^{c/k^2}/k)$ for some constant c. This implies that in order to achieve a constant or even polylogarithmic approximation, the running time of any distributed algorithm must be at least $\Omega(\sqrt{\log n/\log\log n})$. Using the notion of locality-preserving reductions, we extend these MVC lower bounds to other network coordination problems, including exact problems such as MIS or maximal matching. For general graphs, this MIS result improves on Linial's $\Omega(\log^* n)$ lower bound for rings. Moreover, our result currently stands—together with Elkin's lower bound on the distributed approximability of MST in the $\mathcal{CONGEST}$ model—as the only *hardness of distributed approximation* results (see [77]).

All our lower bounds hold even in the $\mathcal{LOCAL}$ model and are therefore true consequences of limitations imposed by locality only, not side-effects resulting from aspects such as congestion, asynchrony, collisions and interference, or bounded message size. The lower bounds are therefore the first general results (for all k) on the achievable trade-off between *locality* and the resulting *approximation quality*. Finally, all lower bounds hold even for randomized algorithms and if nodes have unique identifiers in the range $1,\ldots,n$.

The lower bounds not only limit the achievable performance of any distributed algorithm that is restricted to a specific running time, but they also capture the amount of *information* inherently required by distributed decision makers for solving or approximating global tasks. And because we establish lower bounds for every k, that is, for every size of the local neighborhood around each node, these lower bounds (in combination with the upper bounds of Chapter 6) also characterize the *value of information* of every additional "layer" of neighbors around the decision makers.

The chapter is organized as follows. Section 7.1 presents our main technical result, a general hardness of approximation lower bound for MVC in the $\mathcal{LOCAL}$ model. Using the notion of *locality-preserving reductions*, we then generalize these bounds to various other problems, including exact problems like MIS and maximal matching in subsequent sections. Finally, Section 7.8 shows that simple *capacitated variants* of MVC and MDS are inherently non-local problems and solutions based merely on local information cannot achieve a good approximation ratio.

7.1 General Lower Bound for Vertex Cover

The proofs of our lower bounds are based on the *timeless indistinguishability* argument [92, 157]. In k rounds of communication, a network node can only gather information about nodes which are at most k hops away and hence, only this information can be used to determine the computation's outcome. If we can show that after k communication rounds many nodes see exactly the same graph topology; informally speaking, all these nodes are equally qualified to join the MIS, dominating set, or vertex cover. The challenge is now to construct the graph in such a way that selecting the wrong subset of these nodes is ruinous.

Unfortunately, constructing such a hard graph for the MDS or FL problem for an arbitrary number of communication rounds turns out to be difficult. In order to obtain general lower bounds, we therefore turn our attention to the simpler MVC problem. As mentioned in Section 5.2, the MVC problem has a simple combinatorial structure and, it appears to be an ideal candidate for local computation. Intuitively, a node should be able to decide whether or not to join the vertex cover using information from its local neighborhood only; very distant nodes appear to be superfluous for its decision. The fact that—as discussed in Section 6.1—the aspect of symmetry breaking does not need to be tackled in the MVC problem fuels further hope for efficient local algorithms.

In the sequel, we prove that this intuition is misleading and even such a seemingly simple *local problems* such as MVC cannot be approximated well in a constant number of communication rounds. In other words, we present *hardness of distributed approximation* lower bounds for MVC and related problems that hold even in the $\mathcal{LOCAL}$ model. With this, we show a lower bound on the inherent amount of locality (i.e., information about the topology) needed in order for distributed decision makers to achieve a good approximation to MVC. Interestingly, our lower bounds even hold for *randomized algorithms* as well as for the *fractional* version of MVC.

We start with an outline of the proof. The basic idea is to construct a graph $G_k = (V, E)$, for each positive integer k. In G_k, there are many neighboring nodes that see exactly the same topology in their k-hop neighborhood, that is, no distributed algorithm with running time at most k can distinguish between these nodes. Informally speaking, both neighbors are equally qualified to join the vertex cover. However, choosing the wrong neighbors in G_k will be ruinous.

G_k contains a bipartite subgraph S with node set $C_0 \cup C_1$ and edges in $C_0 \times C_1$ as shown in Figure 7.1. Set C_0 consists of n_0 nodes each of which has δ_0 neighbors in C_1. Each of the $n_0 \cdot \frac{\delta_0}{\delta_1}$ nodes in C_1 has δ_1, $\delta_1 > \delta_0$, neighbors in C_0. The goal is to construct G_k in such a way that all nodes in $v \in S$ see the same topology $\mathcal{T}_{v,k}$ within distance k. In a globally optimal solution, all edges of S may be covered by nodes in C_1 and hence, no node in C_0 needs to join the vertex cover. In a local algorithm, however, the decision of whether or not a node joins the vertex cover depends only on its local view, that is, the pair $(\mathcal{T}_{v,k}, \mathcal{L}(\mathcal{T}_{v,k}))$. We show that because adjacent nodes in S see the same $\mathcal{T}_{v,k}$, every algorithm adds a large portion of nodes in C_0 to

its vertex cover in order to end up with a feasible solution. In other words, we construct a graph in which the symmetry between two adjacent nodes cannot be broken within k communication rounds. This yields suboptimal local decisions and hence, a suboptimal approximation ratio. Throughout the proof, C_0 and C_1 denote the two sets of the bipartite subgraph S.

The proof is organized as follows. The structure of G_k is defined in Subsection 7.1.1. In Subsection 7.1.2, we show how G_k can be constructed without small cycles, ensuring that each node sees a tree within distance k. Subsection 7.1.3 proves that adjacent nodes in C_0 and C_1 have the same view $\mathcal{T}_{v,k}$ and finally, Subsection 7.1.4 derives the lower bounds.

7.1.1 The Cluster Tree

The nodes of graph $G_k = (V, E)$ can be grouped into disjoint sets which are linked to each other as bipartite graphs. We call these disjoint sets of nodes *clusters*. The structure of G_k is defined using a directed tree $CT_k = (\mathcal{C}, \mathcal{A})$ with doubly labelled arcs $\ell : \mathcal{A} \to \mathbb{N} \times \mathbb{N}$. We refer to CT_k as the *cluster tree*, because each vertex $C \in \mathcal{C}$ represents a cluster of nodes in G_k. The *size* of a cluster $|C|$ is the number of nodes the cluster contains. An arc $a = (C, D) \in \mathcal{A}$ with $\ell(a) = (\delta_C, \delta_D)$ denotes that the clusters C and D are linked as a bipartite graph, such that each node $u \in C$ has δ_C neighbors in D and each node $v \in D$ has δ_D neighbors in C. It follows that $|C| \cdot \delta_C = |D| \cdot \delta_D$. We call a cluster *leaf-cluster* if it is adjacent to only one other cluster, and we call it *inner-cluster* otherwise.

Definition 7.1. *The cluster tree CT_k is recursively defined as follows:*

$$\begin{aligned} CT_1 &:= (\mathcal{C}_1, \mathcal{A}_1), \quad \mathcal{C}_1 := \{C_0, C_1, C_2, C_3\} \\ \mathcal{A}_1 &:= \{(C_0, C_1), (C_0, C_2), (C_1, C_3)\} \\ \ell(C_0, C_1) &:= (\delta_0, \delta_1), \quad \ell(C_0, C_2) := (\delta_1, \delta_2), \\ \ell(C_1, C_3) &:= (\delta_0, \delta_1) \end{aligned}$$

Given CT_{k-1}, we obtain CT_k in two steps:

- *For each inner-cluster C_i, add a new leaf-cluster C_i' with $\ell(C_i, C_i') := (\delta_k, \delta_{k+1})$.*
- *For each leaf-cluster C_i of CT_{k-1} with $(C_{i'}, C_i) \in \mathcal{A}$ and $\ell(C_{i'}, C_i) = (\delta_p, \delta_{p+1})$, add $k-1$ new leaf-clusters C_j' with $\ell(C_i, C_j') := (\delta_j, \delta_{j+1})$ for $j = 0 \dots k, j \neq p+1$.*

Further, we define $|C_0| = n_0$ for all CT_k.

Figure 7.1 shows CT_2. The shaded subgraph corresponds to CT_1. The labels of each arc $a \in \mathcal{A}$ are of the form $\ell(a) = (\delta_l, \delta_{l+1})$ for some $l \in \{0, \dots, k\}$. Further, setting $|C_0| = n_0$ uniquely determines the size of all other clusters. In order to simplify the upcoming study of the cluster tree, we need two additional definitions. The *level* of a cluster is the distance to C_0 in the cluster tree (cf. Figure 7.1). The *depth* of a cluster C is its distance

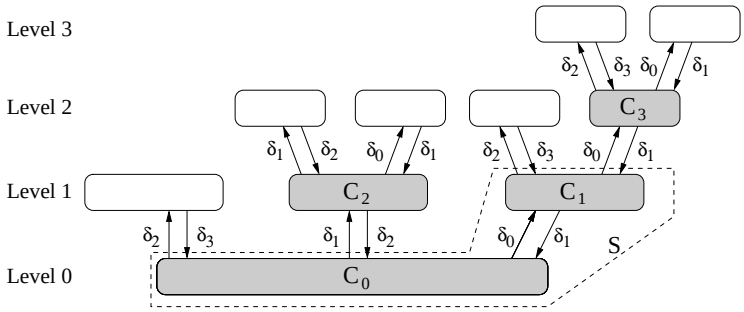

Figure 7.1: Cluster-Tree CT_2.

to the furthest leaf in the subtree rooted at C. Hence, the depth of a cluster plus one equals the height of the subtree corresponding to C. In the example of Figure 7.1, the depths of C_0, C_1, C_2, and C_3 are 3, 2, 1, and 1, respectively.

Note that CT_k describes the general structure of G_k, i.e. it defines for each node the number of neighbors in each cluster. However, CT_k does not specify the actual adjacencies. In the next subsection, we show that G_k can be constructed so that each node's local view is a tree.

7.1.2 The Lower-Bound Graph

In Subsection 7.1.3, we will prove that the topologies seen by nodes in C_0 and C_1 are identical. This task is greatly simplified if each node's topology is a tree (rather than a general graph) because we do not have to worry about cycles. The *girth* of a graph G, denoted by $g(G)$, is the length of the shortest cycle in G. We want to construct G_k with girth at least $2k+1$ so that in k communication rounds, all nodes see a tree. Given the structural complexity of G_k for large k, constructing G_k with large girth is not a trivial task. The solution we present is based on the construction of the graph family $D(r,q)$ as proposed in [158]. For given r and q, $D(r,q)$ defines a bipartite graph with $2q^r$ nodes and girth $g(D(r,q)) \geq r+5$. In particular, we show that for appropriate r and q, we obtain an instance of G_k by deleting some of the edges of $D(r,q)$. In the following, we introduce $D(r,q)$ up to the level of detail which is necessary to understand our results.

For an integer $r \geq 1$ and a prime power q, $D(r,q)$ defines a bipartite graph with node set $P \cup L$ and edges $E_D \subset P \times L$. The nodes of P and L are labelled by the r-vectors over the finite field $\mathbb{F}_q$, i.e. $P = L = \mathbb{F}_q^r$. In accordance with [158], we denote a vector $p \in P$ by (p) and a vector $l \in L$ by $[l]$. The components of (p) and $[l]$ are written as follows (for $D(r,q)$, the

vectors are projected onto the first r coordinates):

$$\begin{aligned}(p) &= (p_1, p_{1,1}, p_{1,2}, p_{2,1}, p_{2,2}, p'_{2,2}, p_{2,3}, p_{3,2}, \dots \\ &\quad p_{i,i}, p'_{i,i}, p_{i,i+1}, p_{i+1,i}, \dots) \qquad (7.1)\\ [l] &= [l_1, l_{1,1}, l_{1,2}, l_{2,1}, l_{2,2}, l'_{2,2}, l_{2,3}, l_{3,2}, \dots \\ &\quad l_{i,i}, l'_{i,i}, l_{i,i+1}, l_{i+1,i}, \dots]. \qquad (7.2)\end{aligned}$$

Note that the somewhat confusing numbering of the components of (p) and $[l]$ is chosen in order to simplify the following system of equations. There is an edge between two nodes (p) and $[l]$, exactly if the first $r-1$ of the following conditions hold (for $i = 2, 3, \dots$).

$$\begin{aligned} l_{1,1} - p_{1,1} &= l_1 p_1 \\ l_{1,2} - p_{1,2} &= l_{1,1} p_1 \\ l_{2,1} - p_{2,1} &= l_1 p_{1,1} \\ l_{i,i} - p_{i,i} &= l_1 p_{i-1,i} \\ l'_{i,i} - p'_{i,i} &= l_{i,i-1} p_1 \\ l_{i,i+1} - p_{i,i+1} &= l_{i,i} p_1 \\ l_{i+1,i} - p_{i+1,i} &= l_1 p'_{i,i} \end{aligned} \qquad (7.3)$$

In [158], it is shown that for odd $r \geq 3$, $D(r,q)$ has girth at least $r+5$. Further, if a node u and a coordinate of a neighbor v are fixed, the remaining coordinates of v are uniquely determined. This is concretized in the next lemma.

Lemma 7.1. *For all $(p) \in P$ and $l_1 \in \mathbb{F}_q$, there is exactly one $[l] \in L$ such that l_1 is the first coordinate of $[l]$ and such that (p) and $[l]$ are connected by an edge in $D(r,q)$. Analogously, if $[l] \in L$ and $p_1 \in \mathbb{F}_q$ are fixed, the neighbor (p) of $[l]$ is uniquely determined.*

Proof. The first $r-1$ equations of (7.3) define a linear system for the unknown coordinates of $[l]$. If the equations and variables are written in the given order, the matrix corresponding to the resulting linear system of equations is a lower triangular matrix with non-zero elements in the diagonal. Hence, the matrix has full rank and by the basic laws of (finite) fields, the solution is unique. Exactly the same argumentation holds for the second claim of the lemma. □

We are now ready to construct G_k with large girth. We start with an arbitrary instance G'_k of the cluster tree which may have the minimum possible girth 4. An elaboration of the construction of G'_k is deferred to Subsection 7.1.4. For now, we simply assume that G'_k exists. Both G_k and G'_k are bipartite graphs with odd-level clusters in one set and even-level clusters in the other. Let m be the number of nodes in the larger of the two partitions of G'_k. We choose q to be the smallest prime power greater than or equal to m. In both partitions $V_1(G'_k)$ and $V_2(G'_k)$ of G'_k, we uniquely label all nodes v with elements $c(v) \in \mathbb{F}_q$.

As already mentioned, G_k is constructed as a subgraph of $D(r,q)$ for appropriate r and q. We choose q as described above and we set $r = 2k-4$

such that $g(D(r,q)) \geq 2k+1$. Let $(p) = (p_1, \dots)$ and $[l] = [l_1, \dots]$ be two nodes of $D(r,q)$. (p) and $[l]$ are connected by an edge in G_k if and only if they are connected in $D(r,q)$ and there is an edge between nodes $u \in V_1(G'_k)$ and $v \in V_2(G'_k)$ for which $c(u) = p_1$ and $c(v) = l_1$. Finally, nodes without incident edges are removed from G_k.

Lemma 7.2. *The graph G_k constructed as described above is a cluster tree with the same degrees δ_i as in G'_k. G_k has at most $2mq^{2k-5}$ nodes and girth at least $2k+1$.*

Proof. The girth directly follows from the construction; removing edges cannot create cycles.

For the degrees between clusters, consider two neighboring clusters $C'_i \subset V_1(G'_k)$ and $C'_j \subset V_2(G'_k)$ in G'_k. In G_k, each node is replaced by q^{2k-5} new nodes. The clusters C_i and C_j consist of all nodes (p) and $[l]$ which have their first coordinates equal to the labels of the nodes in C'_i and C'_j, respectively. Let each node in C'_i have δ_α neighbors in C'_j, and let each node in C'_j have δ_β neighbors in C'_i. By Lemma 7.1, nodes in C_i have δ_α neighbors in C_j and nodes in C_j have δ_β neighbors in C_i, too. □

Remark In [159], it has been shown that $D(r,q)$ is disconnected and consists of at least $q^{\lfloor \frac{r+2}{4} \rfloor}$ isomorphic components which the authors call $CD(r,q)$. Clearly, those components are valid cluster trees as well and we could use one of them for the analysis. As the asymptotic results remain unaffected by this observation, we continue to use G_k as constructed above.

7.1.3 Equality of Views

In this subsection, we prove that two adjacent nodes in clusters C_0 and C_1 have the same *view*, i.e. within distance k, they see exactly the same topology $\mathcal{T}_{v,k}$. Consider a node $v \in G_k$. Given that v's view is a tree, we can derive its *view-tree* by recursively following all neighbors of v. The proof is largely based on the observation that corresponding subtrees occur in both node's view-tree.

Let C_i and C_j be adjacent clusters in CT_k connected by $\ell(C_i, C_j) = (\delta_l, \delta_{l+1})$, i.e. each node in C_i has δ_l neighbors in C_j, and each node in C_j has δ_{l+1} neighbors in C_i. When traversing a node's view-tree, we say that we *enter* cluster C_j (resp., C_i) over *link* δ_l (resp., δ_{l+1}) from cluster C_i (resp., C_j). Furthermore, we make the following definitions:

Definition 7.2. *The following nomenclature refers to subtrees in the view-tree of a node in G_k.*

- *M_i is the subtree seen upon entering cluster C_0 over a link δ_i.*
- *$B_{i,d,\lambda}$ is a subtree seen upon entering a cluster $C \in \mathcal{C} \setminus \{C_0\}$ over a link δ_i, where C is on level λ and has depth d.*

Definition 7.3. *When entering subtree $B_{i,d,\lambda}$ from a cluster on level $\lambda - 1$ ($\lambda + 1$), we write $B^{\uparrow}_{i,d,\lambda}$ ($B^{\downarrow}_{i,d,\lambda}$). The predicate $\neg$ in $B^{\neg}_{i,d,\lambda}$ denotes that instead of δ_i, the label of the link into this subtree is $\delta_i - 1$.*

The predicate $\neg$ is necessary when, after entering C_j from C_i, we immediately return to C_i on link δ_i. In the view-tree, the edge used to enter C_j connects the current subtree to its parent. Thus, this edge is not available anymore and there are only $\delta_i - 1$ edges remaining to return to C_i. The predicates $\uparrow$ and $\downarrow$ describe from which "direction" a cluster has been entered. As the view-trees of nodes in C_0 and C_1 have to be absolutely identical for our proof to work, we must not neglect these admittedly tiresome details.

The following example should clarify the various definitions. Additionally, you may refer to the example of G_3 in Figure 7.2.

Example 7.1. *Consider G_1. Let V_{C_0} and V_{C_1} denote the view-trees of nodes in C_0 and C_1, respectively:*

$$
\begin{array}{lll}
V_{C_0} = B^{\uparrow}_{0,1,1} \cup B^{\uparrow}_{1,0,1} & & V_{C_1} = B^{\uparrow}_{0,0,2} \cup M_1 \\
B^{\uparrow}_{0,1,1} = B^{\uparrow}_{0,0,2} \cup M^{\neg}_1 & & B^{\uparrow}_{0,0,2} = B^{\downarrow,\neg}_{1,1,1} \\
B^{\uparrow}_{1,0,1} = M^{\neg}_2 & & M_1 = B^{\uparrow,\neg}_{0,1,1} \cup B^{\uparrow}_{1,0,1} \\
\dots & & \dots
\end{array}
$$

We start the proof by giving a set of rules which describe the subtrees seen at a given point in the view-tree. We call these rules *derivation rules* because they allow us to *derive* the view-tree of a node by mechanically applying the matching rule for a given subtree.

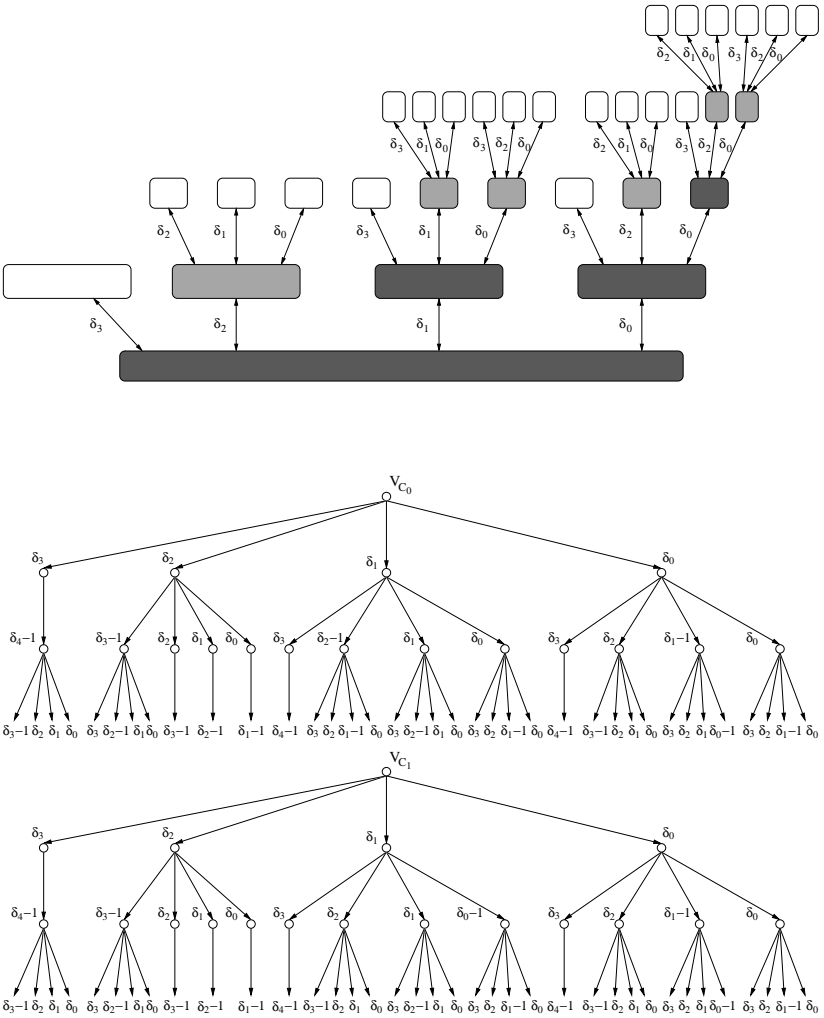

Figure 7.2: The Cluster Tree CT_3 and the corresponding view-trees of nodes in C_0 and C_1. The cluster trees CT_1 and CT_2 are shaded dark and light, respectively. The labels of the arcs of the cluster tree represent the number of higher-level cluster. The labels of the reverse links are omitted. In the view-trees, an arc labelled with δ_i stands for δ_i edges, all connecting to identical subtrees.

Lemma 7.3. *The following derivation rules hold in G_k:*

$$\begin{aligned}
M_i &= \bigcup_{\substack{j=0\ldots k\\ j\neq i-1}} B^{\uparrow}_{j,k-j,1} \cup\ B^{\uparrow,\neg}_{i-1,k-i+1,1}\\
B^{\uparrow}_{i,d,1} &= F_{\{i+1\}} \cup D_{\{\}} \cup\ M^{\neg}_{i+1}\\
B^{\downarrow}_{i,d,1} &= F_{\{i-1,k-d+1\}} \cup D_{\{\}} \cup M_{k-d+1} \cup B^{\uparrow,\neg}_{i-1,d-1,2}\\
B^{\uparrow}_{i,d,\lambda} &= F_{\{i+1\}} \cup D_{\{i+1\}} \cup B^{\downarrow,\neg}_{i+1,d+1,\lambda-1}
\end{aligned}$$

where F and D are defined as

$$\begin{aligned}
F_W &:= \bigcup_{\substack{j=0\ldots k-d+1\\ j\notin W}} B^{\uparrow}_{j,d-1,\lambda+1}\\
D_W &:= \bigcup_{\substack{j=k-d+2\ldots k\\ j\notin W}} B^{\uparrow}_{j,k-j,\lambda+1}.
\end{aligned}$$

Proof. We first show the derivation rule for M_i. It can be seen in Example 7.1 that the rule holds for $k = 1$. For the induction step, we build CT_{k+1} from CT_k as defined in Definition 7.1. $M^{(k)}$ is an inner cluster and therefore, one new cluster $B_{k+1,0,1}$ is added. The depth of all other subtrees increases by 1 and $M^{(k+1)} := \bigcup_{j=0\ldots k+1} B^{\uparrow}_{j,k-j,1}$ follows. If we enter $M^{(k+1)}$ over link δ_i, there will be only $\delta_{i-1} - 1$ edges left to return to the cluster from which we had entered C_0. Consequently, the link δ_{i-1} features the $\neg$ predicate.

The remaining rules follow along the same lines. Let C_i be a cluster with entry-link δ_i which was first created in CT_r, $r < k$. Note that in CT_k, $r = k - d$ holds because each subtree increases its depth by one in each "round". According to the second building rule of Definition 7.1, r new neighboring clusters (subtrees) are created in CT_{r+1}. More precisely, a new cluster is created for all entry-links $\delta_0 \ldots \delta_r$, except δ_i. We call these subtrees *fixed-depth* subtrees F. If the subtree with root C_i has depth d in CT_k, the fixed-depth subtrees have depth $d - 1$. In each $CT_{r'}$, $r' \in \{r + 2, \ldots, k\}$, C_i is an inner-cluster and hence, one new neighboring cluster with entry-link $\delta_{r'}$ is created. We call these subtrees *diminishing-depth* subtrees D. In CT_k, each of these subtrees has grown to depth $k - r'$.

We now turn our attention to the differences between the three rules. They stem from the exceptional treatment of level 1, as well as the predicates $\uparrow$ and $\downarrow$. In $B^{\uparrow}_{i,d,1}$, the link δ_{i+1} returns to C_0, but contains only $\delta_{i+1} - 1$ edges in the view-tree. In $B^{\downarrow}_{i,d,1}$, we have to consider two special cases. The first one is the link to C_0. For a cluster on level 1 with entry-link (from C_0) i, the equality $k = d + i$ holds and therefore, the link to C_0 is δ_{k-d+1} and thus, M_{k-d+1} follows. Secondly, we write $B^{\uparrow,\neg}_{i-1,d-1,2}$, because there is one edge less leading back to the cluster where we had come from. (Note that since we entered the current cluster from a higher level, the link leading back to where we came from is δ_{i-1}, instead of δ_{i+1}). Finally in $B^{\uparrow}_{i,d,\lambda}$, we again have to treat the returning link δ_{i+1} specially.

Note that the general derivation rule for $B^{\downarrow}_{i,d,\lambda}$ is missing as we will not need it for the proof. □

Next, we define the notion of *r-equality*. Intuitively, if two view-trees are r-equal, they have the same topology within distance r.

Definition 7.4. *Let $V_1 = \bigcup_{i=0\ldots k} b_i$ and $V_2 = \bigcup_{i=0\ldots k} b'_i$ be view-trees; b_i and b'_i are subtrees entered on link δ_i. Then, V_1 and V_2 are r-equal if all corresponding subtrees are $(r-1)$-equal,*

$$V_1 \stackrel{r}{=} V_2 \iff b_i \stackrel{r-1}{=} b'_i \ , \ \forall i \in \{0, \ldots, k\}.$$

Further, all subtrees are 0-equal: $B_{i,d,\lambda} \stackrel{0}{=} B_{i',d',\lambda'}$ and $B_{i,d,\lambda} \stackrel{0}{=} M_{i'}$ for all i, i', d, d', λ, and λ'.

Using the notion of r-equality, we can now define what we actually have to prove. We will show that in G_k, $V_{C_0} \stackrel{k}{=} V_{C_1}$ holds. This is equivalent to showing that each node in C_0 sees exactly the same topology within distance k as its neighbor in C_1. We first establish several helper lemmas.

Lemma 7.4. *Let β and β' be sets of subtrees, and let $V_{v_1} = B^{\uparrow}_{i,d,x} \cup \beta$ and $V_{v_2} = B^{\uparrow}_{i,d,y} \cup \beta'$ be two view-trees. Then, for all x and y,*

$$V_{v_1} \stackrel{r}{=} V_{v_2} \iff \beta \stackrel{r-1}{=} \beta'.$$

Proof. Assume that the roots of the subtree of V_{v_1} and V_{v_2} are C_i and C_j. The subtrees have equal depth and entry-link and they have thus grown identically. Hence, all paths which do not return to clusters C_i and C_j must be identical. Further, consider all paths which, after s hops, return to C_i and C_j over link δ_{i+1}. After these s hops, they return to the original cluster and see views V'_{v_1} and V'_{v_2}, differing from V_{v_1} and V_{v_2} only in the placement of the $\neg$ predicate. This does not affect β and β' and therefore,

$$V_{v_1} \stackrel{r}{=} V_{v_2} \iff V'_{v_1} \stackrel{r-s}{=} V'_{v_2} \ \wedge \ \beta \stackrel{r-1}{=} \beta' \ , \ s > 1.$$

The same argument can be repeated until $r - s = 0$ and because $V'_{v_1} \stackrel{0}{=} V'_{v_2}$, the lemma follows. □

Lemma 7.5. *Let β and β' be sets of subtrees, and let $V_{v_1} = B^{\uparrow}_{i,d,x} \cup \beta$ and $V_{v_2} = B^{\uparrow}_{i,d',y} \cup \beta'$ be two view-trees. Then, for all x and y, and for all $r \leq \min(d, d')$,*

$$V_{v_1} \stackrel{r}{=} V_{v_2} \iff \beta \stackrel{r-1}{=} \beta'.$$

Proof. W.l.o.g, we assume $d' \leq d$. In the construction process of G_k, the root clusters of V_{v_1} and V_{v_2} have been created in steps $k - d$ and $k - d'$, respectively. By Definition 7.1, all subtrees with depth $d^* < d'$ have grown identically in both views. The remaining subtrees of V_{v_2} were all created in step $k - d' + 1$ and have depth $d' - 1$. The corresponding subtrees in V_{v_1} have

at least the same depth and hence, each pair of corresponding subtrees are $(d'-1)$-equal. It follows that for $r \leq \min(d, d')$, the subtrees $B^{\uparrow}_{i,d,x}$ and $B^{\uparrow}_{i,d',y}$ are identical within distance r. Using the same argument as in Lemma 7.4 concludes the proof. □

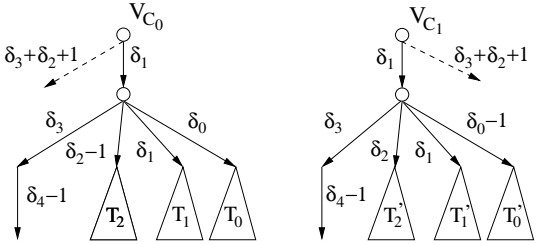

Figure 7.3: The view-trees V_{C_0} and V_{C_1} in G_3 seen upon using link δ_1.

Figure 7.3 shows a part of the view-trees of nodes in C_0 and C_1 in G_3. The figure shows that the subtrees with links δ_0 and δ_2 cannot be matched directly to one another because of the different placement of the -1. It turns out that this inherent difference appears in every step of our theorem. However, the following lemma shows that the subtrees T_0 and T_2 (T'_0 and T'_2) are equal up to the required distance and hence, nodes are unable to distinguish them. It is this crucial property of our cluster tree, which allows us to "move" the $\neg$ predicate between links δ_i and δ_{i+2} and enables us to derive the main theorem.

Lemma 7.6. *Let β and β' be sets of subtrees and let V_{v_1} and V_{v_2} be defined as*

$$\begin{aligned} V_{v_1} &= M_i^{\neg} \cup B^{\uparrow}_{i-2,k-i,2} \cup \beta \\ V_{v_2} &= M_i \cup B^{\uparrow,\neg}_{i-2,k-i,2} \cup \beta'. \end{aligned}$$

Then, for all $i \in \{2, \ldots, k\}$,

$$V_{v_1} \overset{k-i}{=} V_{v_2} \iff \beta \overset{k-i-1}{=} \beta'.$$

Proof. Again, we make use of Lemma 7.3 to show that M_i and $B^{\uparrow}_{i-2,k-i,2}$ are $(k-i-1)$-equal. The claim then follows from the fact that the two subtrees are not distinguishable and the placement of the $\neg$ predicate is irrelevant.

$$M_i = \bigcup_{\substack{j=0\ldots k \\ j\neq i-1}} B^{\uparrow}_{j,k-j,1} \cup B^{\uparrow,\neg}_{i-1,k-i+1,1}$$

$$B^{\uparrow}_{i-2,k-i,2} = \bigcup_{\substack{j=0\ldots i+1 \\ j\neq i-1}} B^{\uparrow}_{j,k-i-1,3} \cup \bigcup_{j=i+2\ldots k} B^{\uparrow}_{j,k-j,3} \cup B^{\downarrow,\neg}_{i-1,k-i+1,1}$$

For $j = \{0, \ldots, i-2, i, \ldots, k\}$, all subtrees are equal according to Lemmas 7.4 and 7.5. It remains to be shown that $B^{\uparrow}_{i-1,k-i+1,1} \overset{k-i-2}{=} B^{\downarrow}_{i-1,k-i+1,1}$. For that purpose, we plug $B^{\uparrow}_{i-1,k-i+1,1}$ and $B^{\downarrow}_{i-1,k-i+1,1}$ into Lemma 7.3 and show their equality using the derivation rules. Let β be defined as $\beta := F_{\{i-2,i\}} \cup D_{\{\}}$.

$$\begin{aligned} B^{\uparrow}_{i-1,k-i+1,1} &= F_{\{i\}} \cup D_{\{\}} \cup M_i^{\neg} \\ &= B^{\uparrow}_{i-2,k-i,2} \cup M_i^{\neg} \cup \beta \\ B^{\downarrow}_{i-1,k-i+1,1} &= F_{\{i-2,i\}} \cup D_{\{\}} \cup M_i \cup B^{\uparrow,\neg}_{i-2,k-i,2} \\ &= B^{\uparrow,\neg}_{i-2,k-i,2} \cup M_i \cup \beta \end{aligned}$$

Again, if M_i and $B^{\uparrow}_{i-2,k-i,2}$ are $(k-i-3)$-equal, we can move the $\neg$ predicate because the subtrees are indistinguishable. Hence, what needs to be shown is $M_i \overset{k-i-3}{=} B^{\uparrow}_{i-2,k-i,2}$. In the proof, we have reduced $V_{v_1} \overset{k-i}{=} V_{v_2}$ stepwise to an expression of diminishing equality conditions, i.e.

$$\begin{aligned} V_{v_1} \overset{k-i}{=} V_{v_2} &\Longleftarrow M_i \overset{k-i-1}{=} B^{\uparrow}_{i-2,k-i,2} \\ &\Longleftarrow B^{\uparrow}_{i-1,k-i+1,1} \overset{k-i-2}{=} B^{\downarrow}_{i-1,k-i+1,1} \\ &\Longleftarrow M_i \overset{k-i-3}{=} B^{\uparrow}_{i-2,k-i,2}. \end{aligned}$$

This process can be continued until either

$$B^{\uparrow}_{i-1,k-i+1,1} \overset{0}{=} B^{\downarrow}_{i-1,k-i+1,1} \quad \text{or} \quad M_i \overset{0}{=} B^{\uparrow}_{i-2,k-i,2}$$

which is always true. □

Finally, we are ready to prove the main theorem.

Theorem 7.7. *Consider graph G_k. Let V_{C_0} and V_{C_1} be the view-trees of two adjacent nodes in clusters C_0 and C_1, respectively. Then, $V_{C_0} \overset{k}{=} V_{C_1}$.*

Proof. Initially, each node in C_0 sees subtree M_* and each node in C_1 sees $B_{*,k,1}$ ($*$ denotes that the subtree has not been entered on any link):

$$\begin{aligned} V_{C_0} &: \quad M_* = \bigcup_{j=0\ldots k} B^{\uparrow}_{j,k-j,1} \\ V_{C_1} &: \quad B_{*,k,1} = \bigcup_{\substack{j=0\ldots k \\ j\neq 1}} B^{\uparrow}_{j,k-j,2} \cup M_1. \end{aligned}$$

It follows $V_{C_0} \overset{k}{=} V_{C_1} \Longleftarrow B^{\uparrow}_{1,k-1,1} \overset{k-1}{=} M_1$ because all other subtrees are $(k-1)$-equal by Lemma 7.4. Having reduced $V_{C_0} \overset{k}{=} V_{C_1}$ to $B^{\uparrow}_{1,k-1,1} \overset{k-1}{=} M_1$,

we can further reduce it to $M_2 \overset{k-2}{=} B^{\uparrow}_{2,k-2,1}$:

$$\begin{aligned} M_1 &= \bigcup_{j=1\ldots k} B^{\uparrow}_{j,k-j,1} \cup B^{\uparrow,\neg}_{0,k,1} \\ B^{\uparrow}_{1,k-1,1} &= B^{\uparrow}_{0,k-2,2} \cup B^{\uparrow}_{1,k-2,2} \cup D_{\{\}} \cup M_2^{\neg} \\ &\underset{\text{Lem. 7.6}}{\overset{k-2}{=}} B^{\uparrow,\neg}_{0,k-2,2} \cup B^{\uparrow}_{1,k-2,2} \cup D_{\{\}} \cup M_2. \end{aligned}$$

By Lemmas 7.4 and 7.5, all subtree are $(k-2)$-equal, except $B^{\uparrow}_{2,k-2,1}$ and M_2.

It seems clear that we can continue to reduce $V_{C_0} \overset{k}{=} V_{C_1}$ step by step in the same fashion until we reach 0. For the induction step, we assume $V_{C_0} \overset{k}{=} V_{C_1} \impliedby B^{\uparrow}_{r,k-r,1} \overset{k-r}{=} M_r$ for $r < k$ and prove $V_{C_0} \overset{k}{=} V_{C_1} \impliedby B^{\uparrow}_{r+1,k-r-1,1} \overset{k-r-1}{=} M_{r+1}$.

$$\begin{aligned} M_r &= \bigcup_{\substack{j=0\ldots k \\ j\neq r-1}} B^{\uparrow}_{j,k-j,1} \cup B^{\uparrow,\neg}_{r-1,k-r+1,1} \\ B^{\uparrow}_{r,k-r,1} &= \bigcup_{j=0\ldots r} B^{\uparrow}_{j,k-r-1,2} \cup D_{\{\}} \cup M^{\neg}_{r+1} \\ &\underset{\text{Lem. 7.6}}{\overset{k-r-1}{=}} \bigcup_{\substack{j=0\ldots r \\ j\neq r-1}} B^{\uparrow}_{j,k-r-1,2} \cup B^{\uparrow,\neg}_{r-1,k-r-1,2} \\ &\quad \cup \bigcup_{j=r+2\ldots k} B^{\uparrow}_{j,k-j,2} \cup M_{r+1}. \end{aligned}$$

Apart from M_{r+1} (resp,. $B^{\uparrow}_{r+1,k-r-1,1}$), all subtrees are $(k-r-1)$-equal by Lemmas 7.4 and 7.5. Since M_{r+1} and $B^{\uparrow}_{r+1,k-r-1,1}$ are the only subtrees not being immediately matched, the induction step follows. For $r = k-1$, we get $V_{C_0} \overset{k}{=} V_{C_1} \impliedby B^{\uparrow}_{k,0,1} \overset{0}{=} M_k$, which concludes the proof because $B^{\uparrow}_{k,0,1} \overset{0}{=} M_k$ is true. □

Remark As a side-effect, the proof of Theorem 7.7 has highlighted the fundamental significance of the *critical path* $P = (\delta_1, \delta_2, \ldots, \delta_k)$ in CT_k. After following path P, the view of a node $v \in C_0$ ends up in the leaf-cluster neighboring C_0 and sees δ_{i+1} neighbors. Following the same path, a node $v' \in C_1$ ends up in C_0 and sees $\sum_{j=0}^{i} \delta_j - 1$ neighbors. There is no way to match these views. This inherent inequality is the underlying reason for the way G_k is defined: It must be ensured that the critical path is at least k hops long.

7.1.4 Analysis

In this subsection, we derive the lower bounds on the approximation ratio of k-local MVC algorithms. Let OPT be an optimal solution for MVC and

let ALG be the solution computed by any algorithm. The main observation is that adjacent nodes in the clusters C_0 and C_1 have the same view and therefore, every algorithm treats nodes in both of the two clusters the same way. Consequently, ALG contains a significant portion of the nodes of C_0, whereas the optimal solution covers the edges between C_0 and C_1 entirely by nodes in C_1.

Lemma 7.8. *Let ALG be the solution of any distributed (randomized) vertex cover algorithm which runs for at most k rounds. When applied to G_k as constructed in Subsection 7.1.2 in the worst case (in expectation), ALG contains at least half of the nodes of C_0.*

Proof. Let $v_0 \in C_0$ and $v_1 \in C_1$ be two arbitrary, adjacent nodes from C_0 and C_1. We first prove the lemma for deterministic algorithms. The decision whether a given node v enters the vertex cover depends solely on the topology $\mathcal{T}_{v,k}$ and the labelling $\mathcal{L}(\mathcal{T}_{v,k})$. Assume that the labelling of the graph is chosen uniformly at random. Further, let $p_0^{\mathcal{A}}$ and $p_1^{\mathcal{A}}$ denote the probabilities that v_0 and v_1, respectively, end up in the vertex cover when a deterministic algorithm $\mathcal{A}$ operates on the randomly chosen labelling. By Theorem 7.7, v_0 and v_1 see the same topologies, that is, $\mathcal{T}_{v_0,k} = \mathcal{T}_{v_1,k}$. With our choice of labels, v_0 and v_1 also see the same distribution on the labellings $\mathcal{L}(\mathcal{T}_{v_0,k})$ and $\mathcal{L}(\mathcal{T}_{v_1,k})$. Therefore it follows that $p_0^{\mathcal{A}} = p_1^{\mathcal{A}}$.

We have chosen v_0 and v_1 such that they are neighbors in G_k. In order to obtain a feasible vertex cover, at least one of the two nodes has to be in it. This implies $p_0^{\mathcal{A}} + p_1^{\mathcal{A}} \geq 1$ and therefore $p_0^{\mathcal{A}} = p_1^{\mathcal{A}} \geq 1/2$. In other words, for all nodes in C_0, the probability to end up in the vertex cover is at least $1/2$. Thus, by the linearity of expectation, at least half of the nodes of C_0 are chosen by algorithm $\mathcal{A}$. Therefore, for every deterministic algorithm $\mathcal{A}$, there is at least one labelling for which at least half of the nodes of C_0 are in the vertex cover.[1]

The argument for randomized algorithms is now straight-forward using Yao's minimax principle. The expected number of nodes chosen by a randomized algorithm cannot be smaller than the expected number of nodes chosen by an optimal deterministic algorithm for an arbitrarily chosen distribution on the labels.

□

Lemma 7.8 gives a lower bound on the number of nodes chosen by any k-local MVC algorithm. In particular, we have that $E[|ALG|] \geq |C_0|/2 = n_0/2$. We do not know OPT, but since the nodes of cluster C_0 are not necessary to obtain a feasible vertex cover, the optimal solution is bounded by $|OPT| \leq n - n_0$. In the following, we define

$$\delta_i := \delta^i \quad , \forall i \in \{0, \ldots, k+1\} \tag{7.4}$$

for some value δ.

[1] In fact, since at most $|C_0|$ such nodes can be in the vertex cover, for at least $1/3$ of the labellings, the number exceeds $|C_0|/2$.

Lemma 7.9. *If $k+1 < \delta$, the number of nodes n of G_k is*

$$n \;\leq\; n_0 \left(1 + \frac{k+1}{\delta - (k+1)}\right).$$

Proof. There are n_0 nodes in C_0. By (7.4), the number of nodes per cluster decreases for each additional level by a factor δ. Hence, a cluster on level l contains n_0/δ^l nodes. By the definition of CT_k, each cluster has at most $k+1$ neighboring clusters on a higher level. Thus, the number of nodes n_l on level l is upper bounded by

$$n_l \;\leq\; (k+1)^l \cdot \frac{n_0}{\delta^l}.$$

Summing up over all levels l and interpreting the sum as a geometric series, we obtain

$$\begin{aligned} n &\leq n_0 \cdot \sum_{i=0}^{k+1} \left(\frac{k+1}{\delta}\right)^l \;\leq\; n_0 \cdot \sum_{i=0}^{\infty} \left(\frac{k+1}{\delta}\right)^l \\ &= n_0 + n_0 \left(\frac{k+1}{\delta}\right)\left(\frac{1}{1-\frac{k+1}{\delta}}\right) \\ &= n_0 \left(1 + \frac{k+1}{\delta-(k+1)}\right). \end{aligned}$$

□

It remains to determine the relationship between δ and n_0 such that G_k can be realized as described in Subsection 7.1.2. There, the construction of G_k with large girth is based on a smaller instance G'_k where girth does not matter. Using (7.4) (i.e. $\delta_i := \delta^i$), we can now tie up this loose end and describe how to obtain G'_k. The number of nodes per cluster decreases by a factor δ on each level of CT_k. Including C_0, CT_k consists of $k+2$ levels. The maximum number of neighbors inside a leaf-cluster is δ^k. Hence, we can set the sizes of the clusters on the outermost level $k+1$ to be δ^k. This implies that the size of a cluster on level l is δ^{2k+1-l}. Particularly, the size of C'_0 at level 0 in G'_k is $n'_0 = \delta^{2k+1}$. Let C_i and C_j be two adjacent clusters with $\ell(C_i, C_j) = (\delta^i, \delta^{i+1})$. C_i and C_j can simply be connected by as many complete bipartite graphs $K_{\delta^i,\delta^{i+1}}$ as necessary.

If we assume that $k+1 \leq \delta/2$, we have $n \leq 2n_0$ by Lemma 7.9. Applying the construction of Subsection 7.1.2, we get $n_0 \leq n'_0 \cdot \langle n' \rangle^{2k-5}$, where $\langle n' \rangle$ denotes the smallest prime power larger than or equal to n', i.e. $\langle n' \rangle < 4n'_0$. Putting all together, we get

$$n_0 \;\leq\; (4n'_0)^{2k-4} \;\leq\; 4^{2k-4}\delta^{4k^2}. \tag{7.5}$$

Theorem 7.10. *There are graphs G, such that in k communication rounds, every distributed algorithm for the minimum vertex cover problem on G has approximation ratios at least*

$$\Omega\left(\frac{n^{c/k^2}}{k}\right) \quad \text{and} \quad \Omega\left(\frac{\Delta^{1/k}}{k}\right)$$

for some constant $c \geq 1/4$, where n and Δ denote the number of nodes and the highest degree in G, respectively.

Proof. We can choose $\delta \geq 4^{-1/(2k)} n_0^{1/(4k^2)}$ due to Inequality (7.5). Finally, using Lemmas 7.8 and 7.9, the approximation ratio α is at least

$$\begin{aligned} \alpha &\geq \frac{n_0/2}{n - n_0} \geq \frac{n_0/2 \cdot \delta/2}{n_0 \cdot (k+1)} = \frac{\delta}{4(k+1)} \\ &\geq \frac{(n/2)^{1/(4k^2)}}{4^{1+1/(2k)}(k+1)} \in \Omega\left(\frac{n^{1/(4k^2)}}{k}\right). \end{aligned}$$

The second lower bound follows from $\Delta = \delta^{k+1}$. □

Theorem 7.11. *In order to obtain a polylogarithmic or even constant approximation ratio, every distributed algorithm for the MVC problem requires at least $\Omega\left(\sqrt{\frac{\log n}{\log\log n}}\right)$ and $\Omega\left(\frac{\log \Delta}{\log\log \Delta}\right)$ communication rounds.*

Proof. We set $k = \beta\sqrt{\log n / \log\log n}$ for an arbitrary constant $\beta > 0$. When plugging this into the first lower bound of Theorem 7.10, we get the following approximation ratio α:

$$\alpha \geq \gamma n^{\frac{c \log\log n}{\beta^2 \log n}} \cdot \frac{1}{\beta}\sqrt{\frac{\log\log n}{\log n}}$$

where γ is the hidden constant in the Ω-notation. For the logarithm of α, we get

$$\begin{aligned} \log \alpha &\geq \frac{c \log\log n}{\beta^2 \log n} \cdot \log n - \frac{1}{2} \cdot \log\log n - \log \beta \\ &= \left(\frac{c}{\beta^2} - \frac{1}{2}\right) \cdot \log\log n - \log \beta. \end{aligned}$$

and therefore

$$\alpha \in \Omega\left((\log n)^{\left(\frac{c}{\beta^2} - \frac{1}{2}\right)}\right).$$

By choosing an appropriate β, we can determine the exponent of the above expression. For every polylogarithmic term $\alpha(n)$, there is a constant β such

that the above expression is at least $\alpha(n)$ and hence, the first lower bound of the theorem follows.

The second lower bound follows from an analogous computation by setting $k = \beta \log \Delta / \log \log \Delta$. □

Remark By defining $\delta_i := \delta^i$, $i \in \{0, \ldots, k\}$ and $\delta_{k+1} := \delta^{k+1/2}$ (instead of δ^{k+1}), we obtain slightly stronger approximation lower bounds of

$$\Omega\left(n^{c/k^2} - k\right) \quad \text{and} \quad \Omega\left(\Delta^{c'/k} - k\right). \tag{7.6}$$

However, these bounds do not suffice to improve the results of Theorem 7.11.

7.2 Locality Preserving Reductions

Using the lower bound for vertex cover, we can obtain lower bounds for several other classical graph problems, including the minimum dominating set and the maximum matching problems. Finally, the *hardness of distributed approximation* lower bound on the MVC problem also gives raise to time lower bounds on the distributed computation of two of the most important exact problems in distributed computing: MIS and maximal matching.

From a more general point of view, the notion of *locality preserving reductions* appears to be interesting by itself. In particular, our reductions could be considered as a first step towards a consistent *classification* of distributed computing problems into some sort of complexity classes according to their *locality*. Ideally, a *theory of locality* based on locality preserving reductions—possibly even including a notion of completeness—could eventually lead to a *hierarchy of locality classes* analogous to the ones found in complexity theory or approximation theory. It would be particularly interesting to establish ties between this hierarchy of locality classes and the classic complexity classes originating in the Turing model of computation. Clearly, the reductions presented in this section fall short of achieving this long-term goal. However, our reductions show that many of the network coordination problems presented in Chapter 5 fall into the same "locality-class" in the sense that they all exhibit the same (or similar) locality-approximation trade-offs. Intriguingly, an analogous statement can also be made for exact problems such as MIS.

7.3 Lower Bounds for MDS and Facility Location

In a non-distributed setting, MDS in equivalent to the general minimum set cover problem, whereas MVC is a special case of set cover which can be approximated much better. It is therefore not surprising that also in a distributed environment, MDS is harder than MVC. In the following, we formalize this intuition giving a *locality-preserving reduction* from MVC to MDS.

Theorem 7.12. *There are graphs G, such that in k communication rounds, every (possibly randomized) distributed algorithm for the minimum dominating set problem on G has approximation ratios at least*

$$\Omega\left(\frac{n^{c/k^2}}{k}\right) \quad \text{and} \quad \Omega\left(\frac{\Delta^{1/k}}{k}\right)$$

for some constant c, where n and Δ denote the number of nodes and the highest degree in G, respectively.

Proof. We show that every MVC instance can be seen as a MDS instance with the same locality. Let $G' = (V', E')$ be a graph for which we want to solve MVC. The corresponding dominating set graph $G = (V, E)$ is constructed as follows. For every node and for every edge in G', there is a node in G. We call nodes $v_n \in V$ corresponding to nodes $v' \in V'$ *n-nodes*, and nodes $v_e \in V$ corresponding to edges $e' \in E'$ *e-nodes*. Two n-nodes are connected by an edge if and only if they are adjacent in G'. An n-node v_n and an e-node v_e are connected exactly if the corresponding node and edge are incident in G'. There are no edges between two e-nodes.

Graphs G' and G exhibit the same localities, i.e. k communication rounds on one of the two graphs can be simulated by $k + \mathrm{O}(1)$ rounds on the other graph. Let C be a feasible vertex cover for G'. We claim that all nodes of G corresponding to nodes in C form a valid dominating set on G. By definition, all e-nodes are covered. The remaining nodes of G are covered because for a given graph, a valid vertex cover is a valid dominating set as well. Therefore, the optimal dominating set on G is at most as big as the optimal vertex cover on G'. There also exists a transformation in the other direction. Let D be a valid dominating set on G. If D contains an e-node v_e, we can replace v_e by one of its two neighbors. The size of D remains the same and all three nodes covered (dominated) by v_e are still covered. By this, we get a dominating set D' which has the same size as D and which consists only of n-nodes. Because D' dominates all e-nodes, the nodes of G' corresponding to D' form a valid vertex cover. Thus, MDS on G is exactly as hard as MVC on G' and the theorem follows from Theorem 7.10. □

Corollary 7.13. *In order obtain a polylogarithmic or constant approximation ratio for minimum dominating set, there are graphs on which every distributed algorithm requires time*

$$\Omega\left(\sqrt{\frac{\log n}{\log\log n}}\right) \quad \text{and} \quad \Omega\left(\frac{\log \Delta}{\log\log \Delta}\right).$$

Proof. The corollary follows directly from Theorem 7.12 and the proof of Theorem 7.11. □

Remark 1 Using the same locality-preserving reduction as from MVC to MDS, it can also be shown that solving the fractional versions of MDS is at least as hard as the fractional version of MVC. Theorem 7.10 also holding for fractional MVC therefore implies that Theorem 7.12 and Corollary 7.13 could equally be stated for fractional MDS.

Remark 2 The minimum dominating set problem is a special case (in which connection costs c_{ij} are 0) of the local facility location problem defined in Chapter 5. The lower bounds of Theorem 7.12 and Corollary 7.13 therefore hold for the facility location problem, as well.

7.4 Lower Bounds for Maximum Matching

While MVC and MDS are standard covering problems, the lower bound can also be extended to *packing problems*. Unfortunately, we are not aware of a simple locality-preserving reduction from MVC to a packing problem. By adjusting the lower-bound graph from Section 7.1.2 somewhat, however, we can show that the same lower bounds hold for the maximum matching problem. In fact, we prove the result for the fractional relaxation of maximum matching in which edges may be selected fractionally, and the sum of these fractional values incident at a single node must not exceed 1. Recall that the *fractional maximum matching problem* is captured by LP_{MM} discussed in Section 5.2, which is the dual to the fractional minimum vertex cover problem. Further, note that $E(v)$ denotes the set of edges incident to node v.

The basic idea of the lower bound follows along the lines of the MVC lower bound in Section 7.1. The view of an edge is defined to be the union of its incident nodes' views. In other words, two edges (u, v) and (u', v') have the same view if $\mathcal{V}_{u,k} \cup \mathcal{V}_{v,k} = \mathcal{V}_{u',k} \cup \mathcal{V}_{v',k}$. The idea is to construct a graph H_k which contains a large set $E' \subset E$ of edges with equal view up to distance k. This implies that, in expectation, the fractional values y_e assigned to the edges in E' must be equal. H_k is constructed in such a way, however, that there are edges in E' who are incident to many other edges in E', whereas the majority of edges in E' are incident to only a few such edges. Clearly, an optimal solution consists of mainly the latter edges, leading to a matching with large cardinality. On the other hand, every distributed k-local algorithm assigns equal fractional values y_e to all edges in E' in expectation. In order to keep the feasibility at the nodes incident to many edges in E', this fractional value must be rather small, which leads to the suboptimality ultimately captured in the theorem.

The construction of H_k uses the lower-bound graph G_k of the MVC lower bound as follows. Let G_k and G'_k be two identical copies of the MVC lower-bound graph defined in Section 7.1. The graph H_k takes the two copies and connects each node in G_k to its counterpart in G'_k as illustrated in Figure 7.4. Formally, let $\phi : V(G_k) \to V(G'_k)$ be an isomorphism mapping nodes of G_k to G'_k. Graph H_k consists of G_k and G'_k, as well as additional edges between nodes $v \in G_k$ and $w \in G'_k$ if and only if $w = \phi(v)$. C'_k denotes the set of

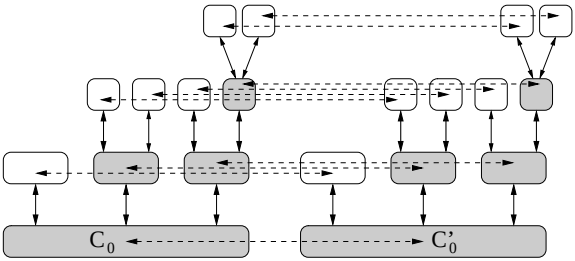

Figure 7.4: The structure of lower-bound graph H_k.

nodes in G'_k corresponding to cluster C_i in G_k. Furthermore, we use the abbreviations $S_0 := C_0 \cup C'_0$ and $S_1 := C_1 \cup C'_1$.

By the construction of H_k and the structural properties proven in Theorem 7.7, the following lemma follows immediately.

Lemma 7.14. *Let v and w be two arbitrary nodes in $S_0 \cup S_1$ of H_k. It holds that v and w see the same topology up to distance k.*

Lemma 7.14 implies that no distributed k-local algorithm can distinguish between edges connecting two nodes in $S_0 \cup S_1$. In particular, this means that edges between C_0 and C_1 cannot be distinguished from edges between C_0 and C'_0. In the sequel, let OPT be the value of the optimal solution for fractional maximum matching and let ALG be the value of the solution computed by any algorithm.

Lemma 7.15. *When applied to H_k, any distributed, possibly randomized algorithm which runs for at most k rounds computes, in expectation, a solution of at most $ALG \leq |S_0|/(2\delta^2) + (|V| - |S_0|)$.*

Proof. First, consider deterministic algorithms. The decision of which value y_e is assigned to edge $e = (v, v)$ depends only on the view the topologies $\mathcal{T}_{u,k}$ and $\mathcal{T}_{v,k}$ and the labelings $\mathcal{L}(\mathcal{T}_{u,k})$ and $\mathcal{L}(\mathcal{T}_{v,k})$, which u and v can collect during the k communication rounds. Assume that the labeling of H_k is chosen uniformly at random. In this case, the labeling $\mathcal{L}(\mathcal{T}_{u,k})$ for any node $u \in V$ is also chosen uniformly at random.

All edges connecting nodes in S_0 and S_1 see the same topology. If the node's labels are distributed uniformly at random, it follows that the *distribution of the views* (and therefore the distribution of the y_e) is the same for all edges connecting nodes in S_0 and S_1. We denote the random variables describing the distribution of the y_e by Y_e. Every node $u \in S_1$ has δ^2 neighbors in S_0. Therefore, for edges e between nodes in S_0 and S_1, it follows by linearity of expectation that $E[Y_e] \leq 1/\delta^2$ because otherwise, there exists at least one labeling for which the computed solution is not feasible. On the other hand, consider an edge e' having both end-points in S_0. By Lemma 7.14, these edges have the same view as edges e between S_0 and S_1.

Hence, for y'_e of e', it must equally hold that $E[Y'_e] \leq 1/\delta^2$. Because there are $|S_0|/2$ such edges, the expected total value contributed to the objective function by edges between two nodes in S_0 is at most $|S_0|/(2\delta^2)$.

Next, consider all edges which do not connect two nodes in S_0. Every such edge has at least one end-point in $V \setminus S_0$. In order to obtain a feasible solution, the total value of all edges incident to a set of nodes V', can be at most $|V'| = |V \setminus S_0|$. This can be seen by considering the dual problem, a kind of minimum vertex cover where some edges only have one incident node. Taking all nodes of V' (assigning 1 to the respective variables) yields a feasible solution for this vertex cover problem. This concludes the proof for deterministic algorithms.

For probabilistic algorithms, we can apply an identical argument based on Yao's minimax principle as in the MVC lower bound (cf Lemma 7.8). □

Lemma 7.15 yields an upper bound on the objective value achieved by any k-local fractional maximum matching algorithm. On the other hand, it is clear that choosing all edges connecting corresponding nodes of G_k and G'_k is feasible and hence, $OPT \geq n/2 \geq |S_0|/2$. Let α denote the approximation ratio achieved by any k-local distributed algorithm, and assume—as in the MVC proof—that $k + 1 \leq \delta/2$. Using the relationship between n, $|S_0|$, δ, and k proven in Lemma 7.9 and combining it with the bound on ALG gives raise to the following theorem.

Theorem 7.16. *There are graphs G, such that in k communication rounds, every (possibly randomized) distributed algorithm for the (fractional) maximum matching problem on G has approximation ratios at least*

$$\Omega\left(\frac{n^{c/k^2}}{k}\right) \quad \textit{and} \quad \Omega\left(\frac{\Delta^{1/k}}{k}\right)$$

for some constant $c \geq 1/4$, where n and Δ denote the number of nodes and the highest degree in G, respectively.

Proof. Using Lemmas 7.15 and 7.9, the approximation ratio α is at least

$$\begin{aligned}\alpha &\geq \frac{|S_0|/2}{\frac{|S_0|}{2\delta^2} + (n - |S_0|)} \underset{\text{Lm 7.9}}{\geq} \frac{|S_0|/2}{\frac{|S_0|}{2\delta^2} + \frac{|S_0|(k+1)}{\delta-(k+1)}} \\ &\geq \frac{1}{\frac{1}{\delta} + \frac{2(k+1)}{\delta}} = \frac{\delta}{4k+5} \in \Omega\left(\frac{n^{1/(4k^2)}}{k}\right).\end{aligned}$$

The second lower bound follows from $\Delta = \delta^{k+2}$. □

Corollary 7.17. *In order to obtain a polylogarithmic or constant approximation ratio, every distributed algorithm for the (fractional) maximum matching problem requires at least*

$$\Omega\left(\sqrt{\frac{\log n}{\log\log n}}\right) \quad \textit{and} \quad \Omega\left(\frac{\log \Delta}{\log\log \Delta}\right)$$

communication rounds.

7.5 Lower Bounds for Maximal Matching

A maximal matching M of a graph G is a maximal set of edges which do not share common end-points. Hence, a maximal matching is a set of non-adjacent edges M of G such that all edges in $E(G) \setminus M$ have a common end-point with an edge in M. The best known lower bound for the distributed computation of a maximal matching is $\Omega(\log^* n)$ which holds for rings [166].

Theorem 7.18. *There are graphs G on which every distributed, possibly randomized algorithm requires time*

$$\Omega\left(\sqrt{\frac{\log n}{\log\log n}}\right) \quad \textit{and} \quad \Omega\left(\frac{\log \Delta}{\log\log \Delta}\right)$$

to compute a maximal matching. This bound holds even if message size is unlimited and nodes have unique identifiers.

Proof. It is well known that the set of all end-points of the edges of a maximal matching form a 2-approximation for MVC. This simple 2-approximation algorithm is commonly attributed to Gavril and Yannakakis. For deterministic algorithms, the lower bound for the construction of a maximal matching in Theorem 7.18 therefore directly follows from Theorem 7.11.

Generalizing this result to randomized algorithms, however, still requires some work. The problem is that Theorem 7.10 lower bounds the achievable approximation ratio by distributed algorithms whose time complexity is exactly k. That is, it does not provide a lower bound for randomized algorithms whose time complexity is at most k in expectation or with a certain probability. As stated in the theorem, however, we consider distributed algorithms that always compute a feasible solution, i.e., only the time complexity depends on randomness. In other words, Theorem 7.10 yields a bound on Monte Carlo type algorithms, whereas in the case of maximal matching, we are primarily interested in Las Vegas type algorithms.

In order to generalize the theorem to randomized algorithms, we give a transformation from an arbitrary distributed maximal matching algorithm $\mathcal{A}_M$ with expected time complexity T into a distributed vertex cover algorithm $\mathcal{A}_{VC}$ with fixed time complexity $2T + 1$ and expected approximation ratio $\alpha_{VC} \in O(\log \Delta)$.

We first define an algorithm $\mathcal{A}'_{VC}$. In a first phase, $\mathcal{A}'_{VC}$ simulates $\mathcal{A}_M$ for exactly $2T$ rounds. Let $E_M \subseteq E$ be the set of edges selected after these rounds. In the second phase, every node v checks whether it has at most one incident edge in E_{VC}. If a node has more than one incident edge in E_{VC}, it removes all these edges from E_{VC}. Hence, E_{VC} forms a feasible matching, although not necessarily a maximal one.

It follows from Markov's inequality that when running $\mathcal{A}_M$ for $2T$ rounds, the probability for obtaining a feasible maximal matching is at least $1/2$. Therefore, algorithm $\mathcal{A}'_{VC}$ outputs a matching that is maximal with probability at least $1/2$. Let $V_{VC} \subseteq V$ denote the set of all nodes incident to an edge in E_{VC}. A maximal matching being a vertex cover, V_{VC} is a feasible

vertex cover with probability at least 1/2. If not, the construction of $\mathcal{A}'_{VC}$ guarantees that $|V_{VC}|$ is at most twice the size of an optimal vertex cover.

Algorithm $\mathcal{A}_{VC}$ executes $\log\Delta$ independent runs of $\mathcal{A}'_{VC}$ in parallel. The probability for obtaining a feasible vertex cover in at least one of the runs is at least $1-\Delta^{-1}$. Let S be the union of the $\log\Delta$ node sets $|V_{VC}|$ computed in the different runs of $\mathcal{A}'_{VC}$ and let OPT_{VC} be the size of an optimal vertex cover. The size of S is at most $2\log\Delta\cdot OPT_{VC}$ and with probability $1-\Delta^{-1}$, S forms a feasible vertex cover. If S is not a feasible vertex cover, every node incident to an uncovered edge joins S, thus guaranteeing S to be a vertex cover. The expected size of S is

$$E[|S|] \leq \left(1-\frac{1}{\Delta}\right)\cdot 2\log\Delta\cdot OPT_{VC}+\frac{n}{\Delta} < \left(2\log\Delta+1+\frac{1}{n-1}\right)\cdot OPT_{VC},$$

where the second inequality follows from the fact that every node can cover at most Δ edges and there are at least $n-1$ edges, and thus $(n-1)/\Delta \leq OPT_{VC}$.

Based on an algorithm $\mathcal{A}_M$ for maximal matching, we can therefore obtain a logarithmic approximation to MVC using a reduction that preserves locality up to a constant factor. Because the time lower bounds of Theorem 7.11 also holds for polylogarithmic approximation ratios, Theorem 7.18 follows. □

7.6 Lower Bounds for Maximal Independent Set

As in the case of a maximal matching, the best currently known lower bound on the distributed complexity of an MIS has been Linial's $\Omega(\log^* n)$ lower bound. Using a locality-preserving reduction from MM to MIS, we can strengthen this lower bound on general graphs as formalized in the following theorem.

Theorem 7.19. *There are graphs G on which every distributed, possibly randomized algorithm requires time*

$$\Omega\left(\sqrt{\frac{\log n}{\log\log n}}\right) \quad \textit{and} \quad \Omega\left(\frac{\log\Delta}{\log\log\Delta}\right)$$

to compute a maximal independent set (MIS). This bound holds even if message size is unlimited and nodes have unique identifiers.

Proof. For the MIS problem, consider the line graph $L(G_k)$ of G_k. The nodes of a line graph $L(G)$ of G are the edges of G. Two nodes in $L(G)$ are connected by an edge whenever the two corresponding edges in G are incident to the same node. The MM problem on a graph G is equivalent to the MIS problem on $L(G)$. Further, if the real network graph is G, k communication rounds on $L(G)$ can be simulated in $k+\mathrm{O}(1)$ communication rounds on G. Therefore, the times t to compute an MIS on $L(G_k)$ and t' to compute a MM on G_k can only differ by a constant, $t \geq t'-O(1)$. Let n' and Δ' denote the number of nodes and the maximum degree of G_k, respectively. The number

of nodes n of $L(G_k)$ is less than $n'^2/2$, the maximum degree Δ of G_k is less than $2\Delta'$. Because n' only appears as $\log n'$, the power of 2 does not hurt and the theorem holds ($\log n = \Theta(\log n')$). □

7.7 Discussion

It is interesting to discuss the lower bounds in relation to the best known upper bounds for the various problems. The MVC algorithm presented in Section 6.1 achieves an $O(\Delta^{1/k})$ approximation in k communication rounds for arbitrary k. For all values of k in $O(\log\Delta/\log\log\Delta)$, it holds that $\Delta^{1/k}/k = \Delta^{1/k'}$ for some $k' \in \Theta(k)$, and hence, the upper and lower bounds achieved in Theorems 6.3 and 7.10 are *tight*. In particular, our MVC algorithm requires $O(\log\Delta/\log\log\Delta)$ communication rounds in order to achieve a polylogarithmic approximation ratio, which is asymptotically optimal. When it comes to constant factor approximations, the algorithm requires time $O(\log\Delta)$, and hence, there is a gap between upper and lower bound of $O(\log\log\Delta)$. It should also be noted that while the lower bounds hold in the $\mathcal{LOCAL}$ model, the MVC algorithm of Section 6.1 works in even in the $\mathcal{CONGEST}$ model. This yields the interesting result that for $k \in O(\log\Delta/\log\log\Delta)$, the distributed complexity of MVC in the $\mathcal{LOCAL}$ and $\mathcal{CONGEST}$ models are equivalent.

Unfortunately, our bounds are not equally tight when expressed as a function of n, rather than Δ. In particular, the gap between upper and lower bound can be as large as $\Theta(\sqrt{\log n/\log\log n})$ for polylogarithmic and $\Theta(\sqrt{\log n\cdot\log\log n})$ for constant approximations, respectively. The additional square-root in the lower bounds when formulated as a function of n follows inevitably from our high girth construction of G_k: In order to derive a lower-bound graph as described in Section 7.1, there must be many "bad" nodes that have the same view as a few neighboring "good" nodes. If each bad node has a degree of δ_{bad} (in G_k, this degree was $\delta_{bad} \in \Theta(n^{1/k^2})$) and if we want to have girth at least k, the graph must consist of at least $n \geq \delta_{bad}^k$ nodes. If we now take all good nodes and apply Algorithm 6.1 of Section 6.1 to the set of bad nodes, we obtain an approximation ratio of $\alpha \in O(\delta_{bad}^{1/k})$ in k communication rounds. Combining this with the bound on the number of nodes in the graph, it follows that we cannot hope for a better lower bound than $\Omega(n^{1/k^2})$ with this technique. From this it follows that if we want to improve the lower bound (i.e., by getting rid of its square-root), we either need an entirely different proof technique, or we must handle graphs with low girth in which nodes do not see trees in their k-hop neighborhood, which would necessitate arguing about views containing cycles.

In comparison to MVC, the lower bounds for (fractional) MDS and facility location deviate more from the currently best known upper bounds, particularly in the $\mathcal{CONGEST}$ model. It is an interesting open question whether a lower-bound graph specifically defined for these problems can give stronger bounds for these problems. Finally, as far as the bounds on MIS and maximal matching are concerned, the lower bounds prove that the classic

randomized algorithms of [9, 169, 129] cannot be improved drastically.

The lower-bound graph G_k of Section 7.1 is a carefully constructed, recursively self-similar graph, which has of course little in common with the network topologies as they are typically encountered in reality. In Chapter 8, we will see that there exist algorithms for more "realistic" classes of network graphs that are faster than the general lower bounds obtained in this chapter, thus clearly separating the two models with regard to their locality.

7.8 Lower Bounds for Capacitated Problems

So far, we have studied hardness of distributed approximation lower bounds for pure covering and packing optimization problems. For certain applications and, generally, from the point of view of distributed approximability, it may also be interesting to understand the locality of extensions or variants of these basic problems. One particularly well-known and important such variant are *capacitated* or *balanced* covering problems. In the *capacitated minimum dominating set problem* (CMDS), every node can cover only a certain number of neighboring nodes. The task is to select a subset $S \subseteq V$ of dominators and to assign each of the remaining nodes $V \setminus S$ to one of the dominators. Formally, let $cap(v) \geq 1$ be the capacity of node v and let $\phi(v)$ denote the dominator $\phi(v) \in S$ that is assigned to node $v \in V$. The *assignment pair* (V, ϕ) determines a solution to the CMDS problem. The assignment is feasible if $\phi(v) \in \Gamma(v)$ and $|\{u|\phi(u) = v\}| \leq cap(v)$ for all $v \in V$.

CMDS and other capacitated covering problems arise naturally in numerous practical settings, because, typically, a leader or center in a network cannot handle an infinitely large number of clients or slave-nodes. A distributed time lower bound for the *capacitated vertex cover problem* was formally proven in [111]. Intuitively, the inherent non-locality of the capacitated vertex cover problem can be understood on a simple ring network. If every node has a capacity of 1 on a ring, all nodes in the ring have to decide on a common direction in order to be able to cover all edges. However, finding such a common direction requires knowledge about the entire ring. This is the same argument used when showing that a ring with an even number of nodes cannot be 2-colored without seeing the whole ring [166].

Whereas the non-locality of capacitated vertex cover is thus clear, MCDS (at least when assuming that each node's capacity is at least 1) appears to be more local, because every node is capable of covering itself. Centralized approximation algorithms for the capacitated dominating set and the non-metric capacitated facility location problem were given by Bar-Ilan, Kortsarz, and Peleg in [28] and [29], respectively. More specifically, [28] presents approximation algorithms for a variety of NP-hard capacitated network center allocation problems. For the capacitated dominating set problem with uniform capacities, they give a beautiful greedy algorithm, which (unless $P = NP$) achieves an asymptotically optimal approximation ratio of $\ln n$. An $O(\log n + \log \rho)$ approximation algorithm for the capacitated non-metric facility location problem, where ρ denotes the largest weight is given in [29].

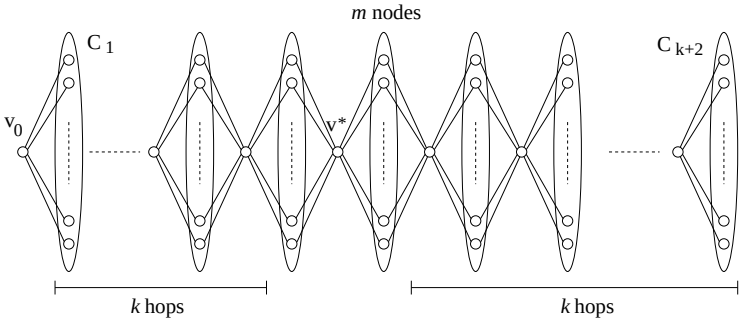

Figure 7.5: The structure of lower-bound graph I_k.

As shown in the following theorem, the capacitated dominating set is non-local, too, even if capacities are uniform.

Theorem 7.20. *There are graphs G, such that in k communication rounds, every (possibly randomized) distributed algorithm for the minimum capacitated dominating set problem on G has approximation ratios at least*

$$\Omega\left(\frac{n}{k^2}\right) \quad \textit{and} \quad \Omega\left(\frac{\Delta}{k}\right)$$

for some constant c, where n and Δ denote the number of nodes and the highest degree in G, respectively. This holds even if capacities are uniform.

Proof. For every $k > 0$, we construct a graph I_k as illustrated in Figure 7.5. We assume for ease of presentation that k is even, the case where k is odd is analogous. I_k is defined as follows. The node set is partitioned into $k+2$ clusters $C_1, \ldots, C_{k+2}$ each containing m nodes. Additionally, there are $k+1$ *connecting nodes* $v_1, \ldots, v_{k+1}$. There is an edge between a connecting node v_i and every node in C_i and C_{i+1}. Finally, there is a designated connecting node v_0 that has a link to either all nodes in C_1 or all nodes in C_{k+2}. Let the capacity of all nodes $v \in V$ be $cap(v) = m + 1$.

Let v^* denote the connecting node $v_{k/2+1}$ in the middle of the graph. After communicating for k rounds, every node has only knowledge about its k-hop neighborhood. By the definition of I_k, neither v^* nor any of its neighbors knows the location of v_0. Therefore, the decision of which node v^* covers cannot depend on the location of v_0.

Consider the nodes that are covered by v^*. Because $cap(v^*) = m + 1$, at most $m/2$ nodes are covered by v^* in either $C_{k/2+1}$ or $C_{k/2+2}$. Without loss of generality, assume that $C_{k/2+2}$ is the cluster in which v^* covers at most $m/2$ nodes. Assume that v_0 is connected to cluster C_1 as in Figure 7.5. There are at least $(k/2+1)m$ nodes to the right of v^* that must be covered. On the other hand, however, there are only $k/2$ connecting nodes v_j for $j > k/2+1$,

each of which can cover at most $m+1$ nodes. The total number of nodes to the right of v^* that can be covered by connecting nodes is therefore at most $k/2 \cdot (m+1) + m/2$. From this, it follows that at least $(m-k)/2$ nodes in these clusters must cover themselves and hence, $ALG \geq (m-k)/2$. On the other hand, the optimal solution can cover all nodes using only connecting nodes when each connecting node v_i covers itself and all nodes in C_{i+1}. Because the total number of nodes is $n = (m+1)(k+2)$, the approximation ratio α of every k-local distributed algorithm is therefore at least

$$\alpha \;\geq\; \frac{(m-k)/2}{k+2} \;=\; \frac{\frac{n}{k+2} - k - 1}{2(k+2)} \;\in\; \Omega\left(\frac{n}{k^2}\right).$$

The second bound follows analogously because of $\Delta = 2m$. □

Note that if we allow *non-uniform capacities*, we can construct a similar lower-bound graph even in simple geometric settings, such as the unit disk graph in an Euclidean plane (see Chapter 8). In particular, every cluster is collapsed to a clique, and bridge-nodes cover all nodes in their neighboring cliques. Every node in a clique has capacity 1, whereas the capacity of bridge-nodes remains $m+1$. On the other hand, it can be shown that in the case of *uniform capacities*, there exist efficient local constant-approximation algorithms in unit disk graphs.

Chapter 8

Locality in Graphs with Bounded Independence or Low Doubling Dimension

In the previous chapters, we have obtained upper and lower bounds on the locality of network coordination problems. The locality lower bounds in turn gave raise to time lower bounds and hardness of approximation results for distributed algorithms. One of the questions arising from these results is their implication on real-world networks. After all, the topology of typical networks are unlikely to look like the lower-bound graph constructed in the previous chapter. The question therefore is, what kind of graphs actually require the amount of locality to be as large as $\Omega(\sqrt{\log n / \log\log n})$ in order to compute an MIS or a constant approximation to MVC or MDS? Or more generally, which fundamental factors of a graph topology (besides n and Δ) determine the strength of the impossibility results?

In this chapter, we aim at answering these questions by looking at the distributed complexity and locality of network coordination problems in graphs that more closely reflect the network topologies appearing in real life. In particular, we are interested in establishing a more fine-grained understanding of the relationship between the underlying network graph and the amount of local knowledge required to solve the distributed coordination tasks discussed in Chapter 5.

Starting from the notion of *unit disk graphs*, we will define the more general family of *graphs with bounded independence* in Section 8.1. These graphs capture many of the characteristic properties exhibited by real networks, particularly in wireless multi-hop networks. Section 8.2 shows that in graphs with bounded independence, an MIS as well as a $(O(1), O(1))$-decomposition can be computed with a deterministic distributed algorithm in time $O(\log\Delta \log^* n)$, greatly improving on the fastest known solutions for general graphs. We then show in Section 8.3 how information about

distances between nodes can be exploited to further reduce the amount of locality required. In particular, we present a deterministic decomposition algorithm with an asymptotically optimal running time of $O(\log^* n)$ if nodes can estimate distances to neighboring nodes. In fact, a closer inspection of this algorithm will reveal an intriguing connection between the distributed complexity and recent work on low-dimensional metric spaces. Finally, Section 8.4 presents a *distributed approximation scheme* for MDS and related problems.

8.1 From Unit Disk Graphs to Graphs of Bounded Independence

In order to capture the specific wireless nature of multi-hop radio networks, researchers have frequently adopted *unit disk graphs* [55] and other geometric intersection graphs to model ad hoc and sensor networks.

Definition 8.1 (Unit Disk Graph (UDG)). *Let $V \subset \mathbb{R}^2$ be a set of nodes in the 2-dimensional Euclidean plane. In a unit disk graph $G = (V, E)$, there is an edge between two nodes $v_1, v_2 \in V$ if and only if the mutual distance is at most 1.*

While modeling wireless multi-hop networks as general graphs may be overly pessimistic, the unit disk graph abstraction is an evident simplification of reality. On the one hand, signal propagation may not form a clear-cut disk and even in homogeneous networks, the model does not account for obstacles which may obstruct signal propagation. In order to address these shortcomings while maintaining some aspects of wireless networks, unit disk graphs can be extended in several directions. In this section, we consider two such possibilities. First, we generalize the underlying metric space and replace the 2-dimensional Euclidean plane by *more general metric spaces*. This naturally leads to the notion of a *unit ball graph* (UBG), which we define as follows.

Definition 8.2 (Unit Ball Graph (UBG)). *Let $M = (X, d)$ be a finite metric space with $n = |X|$ points and distance function $d : X \times X \to \mathbb{R}_0^+$. The graph $G = (V, E)$ with $V = X$ and edge set $E = \{(u, v) \in V \times V \mid d(u, v) \leq 1\}$ is called a* unit ball graph *induced by M.*

Clearly, the unit ball graph induced by the 2-dimensional Euclidean plane is the unit disk graph. In general, we show in Section 8.3 that the complexity of distributed computing tasks in unit ball graphs not only depends on the parameters n and Δ (as shown in Chapters 6 and 7), but also crucially on the *doubling dimension* of the underlying metric space. In [115], the doubling dimension of a metric space is defined as the smallest $\alpha > 0$ such that every ball of radius $2r$ can be covered by 2^α balls of radius r. If α is a constant, the given metric is called *doubling*. Analogously, we call a unit ball graph *doubling* if the underlying metric space is doubling.

In recent years, studying metrics with low doubling dimension has proven to be fruitful in various areas of computer science. For instance, it was

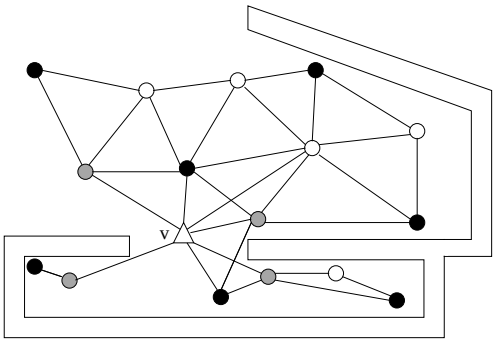

Figure 8.1: A bounded independence graph G with $f(1) = 4$ and $f(2) = 7$. No node u in G has more than 4 and 7 mutually independent nodes in $\Gamma(u)$ and $\Gamma_2(u)$, respectively. For instance, node v has 4 independent neighbors (grey) and 7 independent nodes in $\Gamma_2(v)$ (black). Note that this network can easily be modeled as a BIG even though it looks different from a UDG.

shown that considerably better solutions can be found to problems such as metric space embedding [115], nearest neighbor search [146], approximation algorithms [221], compact routing [3, 40, 217], or data gathering [133], if the underlying metric has low doubling dimension. In Section 8.3, we extend this list by showing that the doubling dimension of a metric space also has a direct influence on the distributed complexity and locality of local problems such as MIS or MDS.

A second way of generalizing the rigid UDG definition is to altogether abandon the notion of an underlying geometry or metric space. This is motivated by the fact that many fundamental results about unit disk graphs do not rely on the actual geometry of unit disk graphs. Consider for instance the classic result that unlike in general graphs, any MIS is a 5-approximation to the MDS problem in a UDG [173].[1] The proof is simple and says that for every node v in the optimal dominating set, the algorithm can select at most 5 MIS nodes in $\Gamma(v)$. The reason is that if we put 6 nodes within distance 1 of v, at least two of these 6 nodes must be within mutual communication range. In other words, this as well as many other results on unit disk graphs only rely on the fact that UDG's are $K_{1,6}$ free, that is, no node has more than 5 mutually non-adjacent neighbors.

Based on the observation that wireless network topologies are $K_{1,\ell}$ free for some constant ℓ, we can formulate the bounded independence model (BIG), which restricts the number of mutually independent nodes within a certain neighborhood of any node $v \in V$. As illustrated in Figure 8.1, this more general model captures the intuitive notion that in wireless networks, nearby nodes tend to hear each other, whereas far-away nodes cannot communicate

[1] Actually, better bounds can be proven. See for instance [95].

because with the available power, radio signals can only be transmitted up to some distance. In other words, if many nodes of a wireless network are located in physical proximity, many of them must be within mutual transmission range. While the disk-shape of the unit disk graph model is easily invalidated by obstacles (such as a wall or a building) or irregular signal propagation, these aspects do not destroy the underlying graph's $K_{1,\ell}$ freeness. Potential, the existence of walls or other obstacles may increase the constant ℓ somewhat, but in all practical settings, the maximal number of mutually independent nodes in a neighborhood of a node is still bounded by a (possibly somewhat larger) constant. This motivates the following definition.

Definition 8.3 (Bounded Independence Graph (BIG)). *A graph G is called* f-independence-bounded *if there is a function $f(r)$ such that every r-neighborhood $\Gamma_r(v)$ of G contains at most $f(r)$ independent (i.e., pairwise non-adjacent) nodes. A graph G has* polynomially bounded independence *if $f(r)$ is a polynomial in r.*

Note that $f(r)$ does not depend on the number of nodes n or any other property of G. Hence, for constant r, the number of independent nodes in an r-neighborhood is constant. Notice that an f-independence bounded graph is $K_{1,f(1)+1}$-free.

Clearly, the unit disk graph is a special case of a bounded independence graph with $f(1) = 5$ and $f(2) \leq 18$. In general, the function f for UDGs can be determined as follows. Consider a node v. The mutual distance between independent nodes in $\Gamma_r(v)$ must be at least 1 and hence, disks of radius $1/2$ around each independent node do not overlap. Moreover, all these disks fit entirely in the disk with radius $r + 1/2$ centered at v. By a standard area argument, it follows that for unit disk graphs

$$f_{UDG}(r) \;\leq\; \frac{\left(r+\frac{1}{2}\right)^2\pi}{\pi/4} \;=\; 4\left(r+\frac{1}{2}\right)^2.$$

For general d-dimensional Euclidean spaces, the corresponding independence function is $f(r) \in O(r^d)$ and a constant degree graph with maximum degree Δ has an independence function of $f(r) \in O(\Delta^r)$.

Properties

In this section, we derive simple properties of *graphs with bounded independence* and *doubling unit ball graphs*. The first lemma shows that if the underlying metric space of a UBG G has constant doubling dimension, G is polynomially independence bounded

Lemma 8.1. *Let $G = (V, E)$ be a UBG induced by a metric space M with doubling dimension α. It holds that G is f-independence-bounded for $f(r) \in O(r^\alpha)$, i.e., there are at most $f(r)$ independent nodes in $\Gamma_r(v)$ for every $v \in V$.*

Proof. In a UBG, the r-neighborhood of node v in G is completely covered by the ball $B_r(v)$ with radius r around v. By the definition of the doubling dimension α, $B_r(v)$ can be covered by at most $2^{\alpha(1+\log r)}$ balls of radius $1/2$. By the triangle inequality, two nodes inside a ball of radius $1/2$ have distance at most 1, that is, the nodes inside a ball of radius $1/2$ form a clique in G. The number of independent nodes in the r-neighborhood of v is therefore at most $2^{\alpha(1+\log r)} \in O(r^\alpha)$. □

Another important property of graphs with bounded independence is that they allow for very efficient *partitions* and *decompositions*. In [23], it is shown that every general graph allows a *sparse cover*, i.e., a clustering of nodes such that every node is in at most $O(\log n)$ clusters and the diameter of every cluster is at most $O(\log n)$. Similarly, we have seen in Section 5.5 that every graph allows for an $(O(\log n), O(\log n))$-decomposition [167]. In the sequel, we show that graphs with bounded independence allow for partitions in which each cluster has constant diameter and the corresponding cluster-graph has constant degree. First, we need the following definition from [196].

Definition 8.4. ***(r-ruling set)*** *Let $S \subseteq V$ be a subset of the nodes of a graph $G = (V, E)$. S is called* r-ruling *if for each node $u \in V \setminus S$, the distance to the closest node in S is at most r.*

If the set S in the above definition is an independent set, we speak of an *r-ruling independent set*. Note that an MIS is a 1-ruling independent set.

An r-ruling independent set S in a graph G induces a natural *clustering* in which every node in S is a cluster-leader and every other node is assigned to its closest cluster-leader (ties being broken arbitrarily). Let $C(s)$ denote the resulting cluster for some $s \in S$. The following lemma shows that this clustering yields a partition whose quality depends solely on the independence function $f(r)$. As customary (see for instance [196]), we define the *cluster-graph* $\tilde{G}(S) = (S, \tilde{E})$ induced by S as the graph in which every cluster is contracted into a single vertex. Moreover, there is an edge between two clusters if there exists an edge in G between nodes in the two clusters, formally $\tilde{E} = \{(s_1, s_2) \mid s_1, s_2 \in S \wedge G$ contains an edge (u, v) for $u \in C(s_1)$ and $v \in C(s_2).\}$. The *cluster-degree* $\Delta_{\tilde{G}}$ is the maximum degree of the cluster-graph.

Lemma 8.2. *The clustering induced by an r-ruling independent set S yields a partition of $G = (V, E)$ with the following properties.*

- *The cluster-degree is at most $f(2r + 1)$.*
- *The diameter of a cluster is at most $2r$.*

Proof. We start with the bound on the cluster-degree. Consider a cluster-leader $s \in S$ and its induced cluster $C(s)$. For every neighboring cluster $C(s')$, it holds that there exists two nodes $u \in C(s)$ and $v \in C(s')$ such that $(u, v) \in E$. Therefore, $d_G(s, s')$ is at most

$$d_G(s, s') \;\leq\; d_G(s, u) + d_G(v, s') + 1 \;\leq\; 2r + 1,$$

because the distance of any node to its cluster-leader is at most r. Because cluster-leaders form an independent set, there can be at most $f(2r+1)$ cluster-leaders within distance $2r+1$. Finally, the bound on the diameter of each cluster follows from the definition of an r-ruling set. □

Notice that because any graph with maximal degree Δ admits a valid coloring with $\Delta+1$ colors, Lemma 8.2 implies the existence of an $(2r, f(2r+1)+1)$-decomposition in G. For constant r, the clustering induced by the r-ruling independent set therefore implies an $(O(1), O(1))$-decomposition with constant cluster-degree.

Finally, we show that in graphs with bounded independence, an MIS can be converted into a constant-diameter/constant-degree partition in time $O(\log^* n)$, and vice versa. In combination with the $\Omega(\log^* n)$ lower bound by Linial [166], this implies the following lemma.

Lemma 8.3. *In any graph with bounded independence, the distributed time complexity of computing an MIS and an $(O(1), O(1))$-decomposition with constant cluster-degree is equivalent up to constant factors.*

Proof. In Section 5.5, we have discussed a distributed procedure for turning a (d, c)-decomposition into an MIS in time $O(c \cdot d)$. As for the other direction, it follows from Lemma 8.2 that the clustering induced by an MIS yields a cluster-graph with constant degree. On this cluster-graph, a coloring can therefore be computed in time $O(\log^* n)$ using the algorithms of [58, 108, 166]. That is, given an MIS in a graph with bounded independence, an $(O(1), O(1))$-decomposition can be obtained in time $O(\log^* n)$. Finally, the claim is concluded by observing that the $\Omega(\log^* n)$ lower bound for computing an MIS on rings—which are graphs of bounded independence—also implies the same lower bound for an $(O(1), O(1))$-decomposition. □

8.2 Fast Deterministic MIS Computation

The distributed complexity of computing an MIS has been of interesting to the distributed computing community for a long time. In Section 5.4, we have seen that there exist efficient randomized algorithms with running time $O(\log n)$, but no deterministic algorithm with polylogarithmic running time is known. In this section, we present a deterministic distributed algorithm which computes an MIS in time $O(\log \Delta \log^* n)$ in graphs with bounded independence. Note that for $\Delta \in o(n^{1/\log^* n})$, our algorithm is faster than the randomized algorithms of [9, 169]. More importantly, however, our algorithm is deterministic and therefore gives a partial answer to the long-standing open problem of the deterministic distributed complexity of computing an MIS. In particular, our result shows that in graphs with bounded independence, an MIS can be computed deterministicly in almost logarithmic time (for large Δ) and in sub-logarithmic time (for small Δ). This is in contrast to the fastest known MIS algorithm for general graphs that has a running time of $O(n^{O(1)/\sqrt{\log n}})$, which is faster than polynomial in n, but slower than polylogarithmic in n.

```
 1: S := ∅;
 2: b(v) := act;
 3: while b(v) = act do
 4:   if ∃u ∈ Γ(v) : b(u) = act then
 5:     d(v) := min{u ∈ Γ(v) | b(u) = act};
 6:     inform neighbor d(v);
 7:     A_v := {u ∈ Γ(v) | d(u) = v};
 8:     if A_v ≠ ∅ then
 9:       p(v) := select one node from A_v;
10:       inform neighbor p(v)
11:     end if;
12:     B_v := {u ∈ Γ(v) | p(u) = v};
13:     if (A_v = ∅) ∧ (B_v = ∅) then
14:       b(v) := pass
15:     else
16:       construct MIS I on graph Ḡ = (V̄, Ē) with V̄ := {u ∈ V | b(u) =
          act} and Ē := {(u, p(u)) | u ∈ V̄ ∧ A_u ≠ ∅};
17:       if v ∉ I then b(v) := pass fi
18:     end if
19:   else
20:     S := S ∪ {v}; b(v) := pass
21:   end if
22: end while
```

Algorithm 8.1: Computing an IS (code for vertex v)

Finally, it should be noted that because every MIS can be turned into an $(O(1), O(1))$-decomposition with constant cluster-degree (Lemma 8.3), our algorithm also gives raise to efficient deterministic solutions for applications such as spanners or synchronizers (see [196]).

The algorithm consists of three phases, which will be described in Sections 8.2.1, 8.2.2, and 8.2.3, respectively. In the first phase, the algorithm computes a sparse, $O(\log \Delta)$-ruling independent set S of the network graph G in time $O(\log \Delta \cdot \log^* n)$. The second phase turns the sparse set S into a dense independent set S' such that each node v of G has a node $u \in S'$ at distance at most 3, that is, S' is 3-ruling. Finally, the algorithm is concluded in Section 8.2.3 by deriving the MIS from the network decomposition induced by the 3-ruling independent set.

8.2.1 Computing a Sparse Independent Set

The first phase of the MIS construction is a distributed algorithm which locally computes an $O(\log \Delta)$-ruling independent set S for a given undirected independence-bounded graph $G = (V, E)$ in time $O(\log \Delta \log^* n)$. A detailed description of the first phase is given by Algorithm 8.1. Before analyzing the algorithm, we give an informal description of the code.

At the beginning, S is empty and all nodes are active (denoted by the

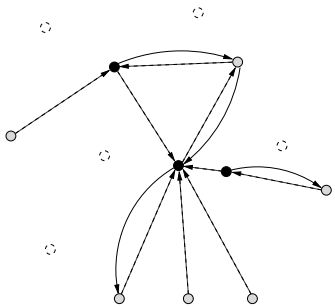

Figure 8.2: One iteration of Algorithm 8.1. The dashed nodes are passive at the outset of the iteration. The dashed arrows between active nodes denote the links $d(v)$. The graph $\overline{G}$ is induced by the links $p(v)$ which are denoted by the solid, bended arrows. Finally, the algorithm computes an MIS on $\overline{G}$, leaving only the black nodes active for the next iteration.

variables $b(v)$ for $v \in V$). Nodes are active as long as they have not decided whether to join the independent set S. As soon as a node becomes passive, it has either joined S in Line 20 or it has decided not to join S. From a general perspective, Algorithm 8.1 tries to eliminate active vertices from the network until single, locally independent nodes are left. It does so with the help of edge-induced subgraphs of bounded degree. In each iteration of the while-loop, a constant-degree graph $\overline{G}$ consisting of active nodes and edges of G is computed. On $\overline{G}$, an MIS can be constructed in time $O(\log^* n)$ [58, 108, 166]. Only the nodes of the MIS of $\overline{G}$ stay active after the iteration of the while-loop. This way, the number of active nodes is reduced by at least a constant factor in every while-loop iteration. As soon as an active node v has no active neighbors, v joins the independent set S (Line 20). The graph $\overline{G}$ is constructed as follows. First, each active node v chooses an active neighbor $d(v)$. Then, each active node u which has been chosen by at least one neighbor v, selects a neighbor $p(u)$ for which $d(p(u)) = u$. The edge set $\overline{E}$ of $\overline{G}$ consists of all edges of the form $(u, p(u))$. Because a node u can only be connected to $d(u)$ and $p(u)$, $\overline{G}$ has at most degree 2. Now, consider a single execution of the while-loop (Lines 3–22, Figure 8.2).

Lemma 8.4. *In the graph $\overline{G} = (\overline{V}, \overline{E})$, every vertex has degree at most 2.*

Proof. Consider $v \in \overline{V}$, then there are at most two vertices adjacent to v by an edge in $\overline{E}$, namely $d(v)$ if $p(d(v)) = v$, and $p(v)$. □

Note that due to this lemma, Line 16 of the algorithm, that is, the local construction of an MIS I on $\overline{G}$, can be completed in $O(\log^* n)$ rounds using methodes described in [58, 108, 166].

Lemma 8.5. *Let V_A denote the set of active nodes. After k iterations of the while-loop, $S \cup V_A$ is a $2k$-ruling set of G.*

Proof. We prove the lemma by induction over the number k of while-loop iterations. Initially all nodes are active, thus the lemma is satisfied for $k = 0$. For the induction step, we show that if a node v becomes passive in an iteration of the while-loop, either v joins S or there is an active node at distance at most 2 from v which remains active for until the next while-loop iteration. Node v can become passive in Lines 14, 17, or 20. If v becomes passive in Line 20, it joins S and therefore the condition of the lemma is satisfied. In Line 17, v is a node of $\overline{G}$ and has a neighbor $u \in \Gamma(v)$ which is in the MIS I of $\overline{G}$. Thus, node u remains active.

The last remaining case is that v decides to become passive in Line 14. By the condition in Line 4, we can assume that v has at least one active neighbor at the beginning of the while-loop iteration. Therefore, v chooses a node $u = d(v)$ in Line 5. Since $A_u \neq \emptyset$, u chooses a node $p(u) \neq v$ and hence, u is a node of $\overline{G}$. Because all nodes of the MIS I of $\overline{G}$ remain active, either u or a neighbor $w \in \Gamma(u) \in \Gamma_2(v)$ is still active after completing the while-loop iteration, which completes the proof. □

The following two lemmas are used to give bounds on the number of rounds needed by Algorithm 8.1 for completion, and to explain the resulting structure in G for general graphs and for independence-bounded graphs, respectively.

Lemma 8.6. *Given an arbitrary graph G, Algorithm 8.1 produces an $O(\log n)$-ruling independent set S after $O(\log n)$ executions of the while-loop.*

Proof. Let n_{act} be the number of active nodes at the beginning of an iteration of the while-loop. We prove that in one while-loop iteration, at least $n_{\text{act}}/3$ nodes become passive. The claim then follows by Lemma 8.5.

Let $\overline{n} \leq n_{\text{act}}$ be the number of nodes of $\overline{G}$ of some particular iteration of the while-loop. All nodes which are not part of $\overline{G}$ become passive in Lines 14 or 20. It therefore suffices to prove that at least one third of the nodes of $\overline{G}$ become passive. $\overline{G}$ is constructed such that it does not contain isolated nodes, that is, all nodes of $\overline{G}$ have at least degree 1. Because the maximum degree of a node in $\overline{G}$ is 2 (Lemma 8.4), the MIS I consists of at most $2\overline{n}/3$ nodes. Hence, at least $\overline{n}/3$ nodes become passive in Line 17. □

The following lemma shows that for independence bounded graphs, the running time can be reduced to $O(\log \Delta)$ (as opposed to $O(\log n)$) executions of the while-loop.

Lemma 8.7. *If the network graph G is independence-bounded, after $O(\log \Delta)$ consecutive execution of the while-loop, Algorithm 8.1 terminates with an $O(\log \Delta)$-ruling independent set S.*

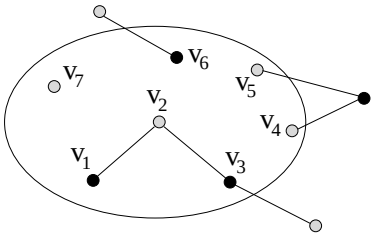

Figure 8.3: The cluster with the edges in $\overline{G}$. Black nodes will remain active in the next iteration. The nodes v_1, v_2, and v_3 are in C_i. Nodes v_4, v_5, and v_6 are connected only to nodes outside of the cluster and hence, are in set C_o. Finally, $v_6 \in C_p$.

Proof. Let M be an MIS of G. The set M defines a clustering of G as follows. We associate a cluster $C(u)$ with each node $u \in M$. Each node $v \notin M$ is assigned to the cluster of an adjacent node $u \in M$. Note that each cluster contains at most $\Delta + 1$ nodes. In the cluster graph $\tilde{G}(M)$, the nodes are the clusters $C(u)$. Two nodes $C(u)$ and $C(v)$ are connected if there is an edge in G connecting the respective clusters. If G is independence bounded, there is a function f such that there are at most $f(3) = O(1)$ independent nodes at distance at most 3 from a node u. Therefore, the maximum degree of $\tilde{G}(M)$ is bounded by $f(3)$.

In the following, we show that the maximum number of active nodes per cluster is reduced by a factor 2 in a constant number of while-loop iterations. For convenience, we define a unit of time to be one iteration of the while-loop. Formally, let α be the maximum number of active nodes per cluster at some time t. We show that there is a constant k such that at time $t+k$ each cluster contains at most $\alpha/2$ active nodes. Note that this implies the lemma because we have $\alpha \leq \Delta+1$ at time $t = 0$. Let $C(u)$ be a cluster with $c > \alpha/2$ active nodes. Consider a single iteration of the while-loop of Algorithm 8.1. We partition the c active nodes of $C(u)$ into three groups according to their neighbors in $\overline{G}$ (Figure 8.3). We denote the set of nodes v which become passive in Line 6 because there is no node w for which $d(w) = u$ by C_p. The set of nodes which have a neighbor inside $C(u)$ and which are only connected to nodes outside $C(u)$ are called C_i and C_o, respectively. Clearly, we have $|C_p| + |C_i| + |C_o| = c$. Because the maximum degree of $\overline{G}$ is 2, at least one third of the nodes in C_i become passive during the MIS construction in Line 10. The nodes in C_o can be divided into the nodes C_o^p which become passive and the nodes C_o^a which stay active. Each node outside $C(u)$ is connected to at most 2 nodes in C_o^a. Therefore, at least $|C_o^a|/2$ nodes outside $\mathcal{C}_u$ become passive. Let $c_i := |C_p| + |C_i| + |C_o^p|$ and $c_o := |C_o^a|$. We have $c_i + c_o = c$. In each iteration of the while-loop at least $c_i/3$ nodes in $C(u)$ and at least $c_o/2$ nodes of clusters which are adjacent to $C(u)$ become passive. Assume that after k iterations of the while-loop, there are still $\alpha/2$ active

Input: t-ruling independent set S
Output: 3-ruling independent set S
1: $S' := S$;
2: **while** S' is not 3-ruling **do**
3: **for each** $u \in S'$ **do**
4: compute $\hat{S}_u \subset \Gamma_4(u)$ such that $S' \cup \hat{S}_u$ is an IS and $\forall v \in \Gamma_3(u), \exists w \in S' \cup \hat{S}_u : \{v, w\} \in E$;
5: $\mathcal{G}$ is the graph induced by $\bigcup_{u \in S'} \hat{S}_u$;
6: $S' := S' \cup \mathrm{MIS}(\mathcal{G})$;
7: **end for**;
8: **end while**

Algorithm 8.2: Computes a dense IS

nodes in $C(u)$. Let $c^{(j)}$, $c_i^{(j)}$, and $c_o^{(j)}$ be the values of c, c_i, and c_o of the j^{th} iteration, respectively. Because there are at most α nodes at the beginning, we have

$$\frac{1}{3} \cdot \sum_{j=1}^{k} c_i^{(j)} \;\leq\; \frac{\alpha}{2} \tag{8.1}$$

because otherwise at least $\alpha/2$ nodes of $C(u)$ would have become passive. Therefore, the number of nodes in the neighboring clusters of $C(u)$ that have become passive is at least

$$\frac{1}{2} \cdot \sum_{j=1}^{k} c_o^{(j)} \;=\; \frac{1}{2} \cdot \sum_{j=1}^{k} \left(c^{(j)} - c_i^{(j)} \right) \;\geq\; \frac{k\alpha}{4} - \frac{1}{2} \cdot \sum_{j=1}^{k} c_i^{(j)}.$$

By Equation (8.1), this is at least $(k-3)\alpha/4$. Because there are at most $d\alpha$ active nodes in neighboring clusters of $C(u)$ at the beginning, after $O(f(3)) = O(1)$ iterations of the while-loop, there are no active nodes in the neighborhood of $C(u)$ left. From then on, at least one third of the nodes in $C(u)$ becomes passive in every further iteration. □

Summarizing the Lemmas 8.4–8.7, we obtain the following theorem.

Theorem 8.8. *Algorithm 8.1 is a local, distributed algorithm which computes an* $\mathrm{O}(\log \Delta)$*-ruling independent set in* $O(\log \Delta \cdot \log^* n)$ *rounds for any independence-bounded graph* $G = (V, E)$*. For general graphs, the Algorithm terminates in* $O(\log n \cdot \log^* n)$ *rounds producing an* $O(\log n)$*-ruling independent set. All messages are of size* $O(\log n)$*.*

8.2.2 From Sparse to Dense: Making the Ruling Independent Set Dense

This section shows how the sparse ruling independent set constructed in the previous section can be made dense enough to achieve an $(O(1), O(1))$-decomposition for graphs with bounded independence. Specifically, we show

how for any integer $t > 3$, a t-ruling independent set can be transformed into a 3-ruling independent set in such graphs in time $O(t \log^* n)$, even in the $\mathcal{CONGEST}$ model. Algorithm 8.2 describes the basic method to achieve this. The idea is to enlarge the independent set iteratively such that it becomes denser in each step. In Line 4, each node of the independent set adds new nodes to the independent set such that each neighbor in distance at most 3 has a neighbor in the extended set. Because every independent set node adds new nodes, it is not guaranteed that the additional nodes generated by different independent set nodes are independent. Therefore, in Lines 5 and 6, the independence of the extended independent set is restored by computing an MIS on the new nodes (see Lemma 8.10). The following lemma shows that in each iteration of the while-loop, the maximum distance of any node to the next node of S' decreases by at least 1.

Lemma 8.9. *Let S' be a t-ruling independent set for $t > 3$. After one iteration of the while-loop of Algorithm 8.2, S' is a $(t-1)$-ruling independent set.*

Proof. We first prove that S' remains an independent set throughout the algorithm. The sets $\hat{S}_u$ are constructed such that nodes in S' and nodes in $\hat{S}_u$ are independent. We therefore only have to prove that all the new nodes form an independent set. This is guaranteed because in Line 6, an MIS of the graph induced by all the new nodes is computed.

To prove that the maximum distance from a node to the next independent set node decreases, consider a node $v \in V$ for which the distance to the nearest node $u \in S'$ is $t > 3$. We prove that after an iteration of the while-loop, the distance between v and the closest node in S' is at most $t-1$. The set $\hat{S}_u$ is constructed such that every node w in the 3-neighborhood $\Gamma_3(u)$ has a neighbor in $S' \cup \hat{S}_u$. On a shortest path (of length t) connecting u and v, let x be the node which is at distance exactly 3 from u. There must be a neighbor y of x for which $y \in \hat{S}_u$. After computing the MIS in Line 6, either y or a neighbor $z \in \Gamma(y)$ joins the independent set S'. The distance between v and y is at least $t-2$ and the distance between v and z is at least $t-1$, which concludes the proof. □

It remains to show that Algorithm 8.2 can indeed be implemented by an efficient distributed algorithm.

Lemma 8.10. *Let G be a graph with bounded independence. On G, Algorithm 8.2 can be executed by a distributed algorithm with time complexity $O(t \log^* n)$ using messages of size $O(\log n)$.*

Proof. By Lemma 8.9, Algorithm 8.2 terminates after at most t iterations of the while-loop. It therefore remains to prove that each while-loop iteration can be exectuted in time $O(\log^* n)$ using messages of size $O(\log n)$. Let us first look at the construction of $\hat{S}_u$ for some node $u \in S'$. A node $v \in \Gamma_4(u)$ can potentially join $\hat{S}_u$ if it has no neighbor in $S' \cup \hat{S}_u$ and if it has an uncovered neighbor $w \in \Gamma_3(u)$, that is, w has no neighbor in $S' \cup \hat{S}_u$. We

call such a node v *candidate*. We add a candidate v to $\hat{S}_u$ if it has a lower ID than all adjacent candidates. Finding out whether a node is a candidate and whether it has the lowest ID among its neighbor candidates can be done in 3 rounds. First, all nodes of $S' \cup \hat{S}_u$ inform their neighbors that they are in the independent set. Then, all covered nodes in $\Gamma_3(u)$ inform their neighbors which can now decide whether they are candidates. Finally, the candidates exchange their IDs. We call those 3 rounds a step. In each step, at least the candidate with the lowest ID joins $\hat{S}_u$. Because G is an independence bounded graph, there can be at most $f(4) = O(1)$ independent nodes in $\Gamma_4(u)$ for some function f. Hence, the number of nodes in $\hat{S}_u$ and therefore the number of steps needed to construct $\hat{S}_u$ is constant. Note that if there was no restriction on the message size, u could collect the complete 4-neighborhood, locally compute $\hat{S}_u$, and inform the nodes in $\hat{S}_u$ in 8 rounds.

It now remains to prove that the construction of the MIS in Line 6 of Algorithm 8.2 can be computed in $O(\log^* n)$ rounds. Consider a node v of the graph $\mathcal{G}$ induced by the union of the sets $\hat{S}_u$ for all $u \in S'$. It holds that $v \in \hat{S}_u$ for some $u \in S'$. Further, let w be a neighbor of v in $\mathcal{G}$. Node w is in $\hat{S}_{u'}$ for some node $u' \in S' \setminus \{u\}$. Because $\hat{S}_{u'}$ consists of nodes of $\Gamma_4(u')$, the distance between v and u' is at most 5. Since G is an independence bounded graph, there exists a function f such that there are at most $f(5)$ independent nodes within distance 5 from v. Thus, there are at most $f(5)$ possible nodes $u' \in S'$ which can cause neighbors w for v. Because all nodes in $\hat{S}_{u'}$ are independent, the number of neighbors of w in $\hat{S}_{u'}$ is at most $f(1)$. Therefore, the maximum degree of the graph $\mathcal{G}$ can be upper bounded by $f(5) \cdot f(1) = O(1)$. On this constant-degree graph, an MIS can be constructed in $O(\log^* n)$ rounds using messages of size $O(\log n)$ [58, 108, 166]. □

Combining Lemmas 8.9 and 8.10 we obtain the next theorem.

Theorem 8.11. *On an independence-bounded graph, a t-ruling independent set can be transformed into a 3-ruling independent set in $O(t \log^* n)$ rounds using messages of size $O(\log n)$.*

8.2.3 Computing the MIS

The algorithm's last phase turns the 3-ruling independent set S' from Algorithm 8.2 into an MIS. By Lemma 8.3, the degree of the cluster-graph $\tilde{G}(S')$ is bounded by $f(7) = O(1)$ if G is f-independence-bounded. The first step of the third phase of our MIS algorithm is to compute $\tilde{G}(S')$ and to color $\tilde{G}(S')$ with $f(7) + 1$ colors, resulting in an $(O(1), O(1))$-decomposition of G. Applying algorithms from [58, 108, 166], this can be achieved in the $\mathcal{CONGEST}$ model in time $O(\log^* n)$.

As already discussed in Section 5.5, this decomposition can be used to determine an MIS M of G by sequentially computing the contributions from each color of the coloring of $\tilde{G}(S')$. For each node v, let $c(v)$ be the color of v's cluster. Using the cluster colors and the node identifiers, we define a

lexicographic order $\prec$ on the set V such that for $u, v \in V$, $u \prec v$ if and only if $c(u) < c(v)$ or if $c(u) = c(v) \wedge \mathrm{ID}(u) < \mathrm{ID}(v)$. Each node now proceeds as follows. Initially, set $M = S'$. All nodes v of S' inform their neighbors about their joining M by sending a JOIN(v) message. If a node u receives a JOIN(v) message from a neighbor v, it cannot join the MIS any more and therefore sends a COVERED(u) message to all neighbors. If a node v has not received a JOIN(u) message but has received a COVERED(u) from all $u \in \Gamma(v)$ for which $u \prec v$, it can safely join M. Note that all neighbors $w \in \Gamma(v)$ with $w \succ v$, would need to receive a COVERED(v) message from v before joining M. If a node v joins M, it informs its neighbors by sending a JOIN(v) message. The described algorithm computes an MIS M in constant time.

Lemma 8.12. *In any f-independence-bounded graph, the above algorithm computes a MIS M in time $2f(7)f(3)$.*

Proof. To see that M is indeed an MIS, consider any two adjacent nodes u and v with $u \prec v$. Assuming that u joins M means that v must have received a COVERED(u) message from u and therefore does not join the MIS itself. Finally, observe that as long as M is not maximal, there is a smallest node u (with respect to $\prec$) which is not covered.

As for the algorithm's time complexity, note that each cluster can contain at most $f(3)$ MIS nodes. Let us now look at a single cluster $C(u)$ of the smallest color 1. Because with respect to $\prec$, the nodes of $C(u)$ are smaller than all nodes of neighboring clusters, the smallest uncovered node of $C(u)$ is always free to join M. When a node v joins M, it takes two rounds until the neighbors of v have forwarded the information that they have been covered. Because at most $f(3)$ nodes of $C(u)$ join M, it takes at most $2f(3)$ rounds until all nodes of color 1 are covered or have joined M. As soon as there is no uncovered node of a color i, the above argument holds for color $i + 1$. Therefore, after at most $f(7) \cdot 2f(3)$ rounds, all nodes are either covered or have joined M. □

In summary, we can combine Theorems 8.8, 8.11, and 8.12 and obtain the following result.

Theorem 8.13. *Let G be a graph of bounded independence. There is a deterministic distributed algorithm which constructs an MIS on G in $O(\log \Delta \cdot \log^* n)$ communication rounds in the $\mathcal{CONGEST}$ model.*

8.3 Faster Algorithms with Coordinate or Distance Information

In the previous section, we have seen that an MIS can be computed in polylogarithmic time by a deterministic distributed algorithm. In this section, we show that even much faster algorithms are possible if nodes have knowledge about their coordinates or their distances to neighbors in a unit ball graph with bounded doubling dimension.

Studying the distributed complexity of network protocols in models in which nodes have more information than mere connectivity (i.e., they do not only know their neighbors) is motivated by practical protocol design for wireless networks. In wireless multi-hop networking, it is often assumed that nodes know their coordinates, or nodes can measure or estimate distances or angles to neighboring nodes by measuring received signal strengths. From a theoretical point of view, this raises numerous interesting questions. In particular, what is the value of such additional information when it comes to distributed computation? Or in other words, how much does this knowledge reduce the required locality of local network problems?

Intuitively, it seems plausible that knowing coordinates or distances allows for faster (and therefore more local) distributed algorithms for many problems. In this section, we formalize this intuition by presenting efficient distributed decomposition algorithms for networks modeled as unit ball graphs. In particular, we show in Section 8.3.1 that computing an $(O(1), O(1))$-decomposition can trivially be computed even without communication if nodes know their coordinates.

The case with known distances discussed in subsequent Section 8.3.2 is more interesting. In particular, we present an algorithm that computes an $(O(1), O(1))$-decomposition in a doubling UBG in asymptotically optimal time $O(\log^* n)$, thus breaking the $\Omega(\sqrt{\log n / \log\log n})$ barrier that holds in general graphs. Interestingly, a closer inspection of this running time reveals a dependency on the doubling dimension of the UBG's underlying metric space. Therefore, this result establishes an intriguing connection between the complexity and locality of distributed computing problems and the doubling dimension of metrics.

8.3.1 Decomposition with Known Coordinates

In many protocols for routing, data gathering, topology control, clustering, or location services in wireless networks, it is implicitly or explicitly assumed that nodes either know their coordinates or–at least–have a rough idea about their coordinates, e.g. [2, 151, 164, 231]. Typically, it is argued that such information can either be obtained from a positioning system such as GPS, or by running a distributed positioning protocol.

In the context of distributed computation, having knowledge about coordinates turns out to be extremely powerful. In particular, knowing coordinates allows to compute an $(O(1), O(1))$-decomposition without any communication at all. Consequently, local network coordination structures such as dominating sets or an MIS can be computed in constant time and locality.

Theorem 8.14. *In a unit disk graph G, nodes are able to compute an $(1, 8)$-decomposition without communication if every node knows its coordinates.*

Proof. Because nodes know their global coordinates, they share a common global coordinate system. The nodes partition the plane by a grid into square cells of side length $1/\sqrt{2}$, each cell defining a cluster. By checking its coordinates, every node can decide to which cluster it belongs. Since the length

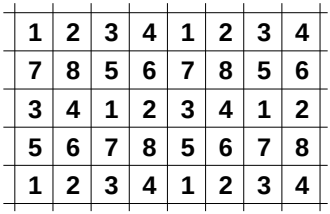

Figure 8.4: Coloring of the grid with 8 colors

of the diagonal of a single square cell is 1, the graph induced by each cluster is a clique. A proper coloring of the cluster graph is obtained by globally coloring the grid such that no two cells whose distance is less than 1 are colored with the same color. Figure 8.4 shows how this can be achieved using 8 colors. Hence, by assigning each cluster the respective color, we obtain a $(1, 8)$-decomposition. □

The possibility of computing an $(O(1), O(1))$-decomposition from UDG coordinates alone indicates the power of coordinate information. Specifically, it implies that coordinate information suffices to compute essentially all local network coordination problems of Chapter 5 in a constant number of rounds.

8.3.2 Network Decomposition with Known Distances

In contrast to coordinate information, it is not intuitively clear how much knowing *distances to neighboring nodes* helps in the distributed computation of independent sets or network decompositions. Clearly, distance information is weaker than coordinate information and in fact, it is not hard to see that it is impossible to construct an MIS or a decomposition in a constant number or rounds, as was the case in Section 8.3.1. In particular, Linial's $\Omega(\log^* n)$ lower bound for computing an MIS in a ring [166] applies to this model because in a ring where all edges have length 1, distance information is useless.

Interestingly, we show in this section that in doubling unit ball graphs, $O(\log^* n)$ rounds indeed suffice to compute an $(O(1), O(1))$-decomposition (and therefore an MIS). More generally, the distributed complexity and the quality of the decomposition depends on the doubling dimension of the underlying metric space. This result also applies in related models where, for instance, the distance between two nodes reflects the propagation delay of messages between two nodes.

Basic Algorithm

We begin by presenting a (potentially slow) deterministic distributed algorithm which computes a partition of the nodes into disjoint clusters, such that the diameter of each cluster is at most 2 and the cluster-graph has constant degree. In a second step, we then provide an efficient implementation

1: $r := \min\{2^{-\lambda} \mid \lambda \in \mathbb{N} \wedge 2^{-\lambda} \geq d_{\min}\}$;
2: $\mathcal{V} := V$;
3: **while** $r \leq 1/2$ **do**
4: $\quad \mathcal{G}_r := (\mathcal{V}, \mathcal{E}_r)$ with $\mathcal{E}_r = \{\{u, v\} \mid d(u, v) < r\}$;
5: $\quad$ compute MIS $S \subseteq \mathcal{V}$ on $\mathcal{G}_r$; [58, 166]
6: $\quad \mathcal{V} := \{v \in \mathcal{V} \mid v \text{ in } S\}$;
7: $\quad r := r \cdot 2$
8: **end while**;
9: All nodes in $\mathcal{V}$ are cluster leaders, the other nodes belong to the cluster of the nearest leader.
10: Let Δ_C be the maximum degree of the cluster graph $\tilde{G}(\mathcal{V})$. Color $\tilde{G}(\mathcal{V})$ with $\Delta_C + 1$ colors. [58, 166]

Algorithm 8.3: Network Decomposition: Clustering

of the algorithm with time complexity $O(\log^* n)$. For the slow version of the algorithm, we assume for ease of presentation that all nodes know the minimum distance $d_{\min}$ between any two nodes. For the fast implementation, the assumption is not required anymore. Finally, let α denote the doubling dimension of the underlying metric space.

Algorithm 8.3 starts with a small radius r which is increased by a factor 2 in every iteration of the while-loop. At the beginning, the set $\mathcal{V}$ of possible cluster leaders contains all nodes. In each iteration, the algorithm selects a subset $S \in \mathcal{V}$, such that the nodes selected in the subset form a maximal independent set on the graph $\mathcal{G}_r = (\mathcal{V}, \mathcal{E}_r)$, where $\mathcal{E}_r$ denotes the set of all edges of length at most r.

Lemma 8.15. *Algorithm 8.3 computes a $(2, 2^{4\alpha})$-decomposition where α is the doubling dimension of the underlying metric. The maximum degree of the cluster graph is at most $2^{4\alpha} - 1$.*

Proof. We first prove the bound on the cluster diameter. The algorithm maintains a set $\mathcal{V}$ of nodes which are candidates for becoming cluster leader. In each iteration, only nodes in MIS S remain in $\mathcal{V}$. We show that for all nodes u which are removed, there is a node v with $d(u, v) \leq 1$ which stays in $\mathcal{V}$ until the end, that is, v becomes cluster leader. Let $r_u = 2^{-\lambda_u}$ ($\lambda_u \in \mathbb{N}$) be the radius at which u is removed from $\mathcal{V}$. Because S is an MIS on $\mathcal{G}_r$, it holds that whenever a node is removed from $\mathcal{V}$, there is a node at distance at most r which stays in $\mathcal{V}$. Hence, after removing u, there is a node $u_0 \in \mathcal{V}$ with $d(u, u_0) \leq r_u$. If u_0 is removed in the subsequent iteration, there is a node u_1 with $d(u_0, u_1) \leq 2r_u$ which remains in $\mathcal{V}$. In general, we get a sequence $u_0, u_1, \ldots, u_i, \ldots$ of nodes where $d(u_{i-1}, u_i) \leq 2^i r_u$ such that u_i remains in $\mathcal{V}$, i iterations after the removal of u. Summing up the distances results in a geometric series. For the distance between u and u_i, it therefore holds

$$d(u, u_i) \;\leq\; \sum_{j=0}^{i} 2^j r_u \;<\; 2^{i+1} r_u \;=\; 2r_{u_i},$$

where r_{u_i} is the radius of the iteration in which node u_i remains in $\mathcal{V}$ and where u_{i-1} is removed from $\mathcal{V}$. Let v be the last node in the sequence, that is, v is a cluster leader. Because the radius of the last iteration of Algorithm 8.3 is $1/2$, we have $d(u, v) < 1$. Thus, the radius of each cluster is at most 1, and hence the diameter of each cluster at most 2.

It now remains to show that the maximum cluster-degree Δ_C of the cluster graph is at most $2^{4\alpha} - 1$. On the one hand, the last iteration of the algorithm ($r = 1/2$) guarantees that the distance between any two cluster leaders is at least $1/2$. Otherwise, the set S computed in Line 5 would not be independent. Consequently, each ball of radius $1/4$ or smaller contains at most one cluster leader. On the other hand, because the radius of each cluster is at most 1, the distance between two cluster leaders of adjacent clusters is at most 3. This means that for a cluster leader v, all leaders of adjacent clusters are located in $B_3(v)$, the ball with radius 3 around v. By the definition of α, $B_3(v)$ can be covered by at most $2^{4\alpha}$ balls of radius $3/16 < 1/4$. Including v, the number of cluster leaders in $B_3(v)$ is therefore at most $2^{4\alpha}$. □

As for the time complexity of a single iteration of the algorithm's while-loop, the dominating factor is the MIS computation in Line 5. Everything else (computing the neighbors in $\mathcal{G}_r$ and informing neighbors about new $\mathcal{V}$) can be done in a constant number of rounds. The time complexity for computing an MIS by a distributed algorithm depends on the maximum degree Δ of the graph. For small Δ, the fastest known algorithms are based on *coloring* algorithms. Because a coloring with K colors is identical to a $(1, K)$-decomposition, we know from Section 5.5 that a K-coloring can be turned into an MIS in K rounds of communication. In other words, a graph can be colored with K colors in t rounds, an MIS can be computed in $t + K$ rounds. In [58], Cole and Vishkin presented an elegant algorithm for coloring a graph with 3^Δ colors in time $O(\log^* n)$, resulting in an MIS algorithm with time complexity $O(\log^* n + 3^\Delta)$. The algorithm was improved to $O(\log \Delta(\Delta^2 + \log^* n))$ by Goldberg, Plotkin, and Shannon in [108]. In [166], Linial shows that any graph can be colored with $O(\Delta^2)$ colors in $O(\log^* n)$ rounds yielding a time complexity for computing an MIS of $O(\Delta^2 + \log^* n)$. For our purposes, the important thing is that for constant Δ, all three algorithms manage to compute an MIS in $O(\log^* n)$ rounds. Lemma 8.16 bounds the maximum degree of $\mathcal{G}_r$.

Lemma 8.16. *In each iteration of Algorithm 8.3, the maximum degree of $\mathcal{G}_r$ is at most $2^{2\alpha}$.*

Proof. Let ℓ be the length of the minimum distance between any two nodes of $\mathcal{G}_r$. Because the algorithm computes an independent set in each iteration, we have $\ell > r/2$. Therefore, every ball of radius $r/4$ contains only one node in $\mathcal{V}$. All neighbors of a node $v \in \mathcal{V}$ in $\mathcal{G}_r$ are located in the ball $B_r(v)$. By the definition of the doubling dimension α, $B_r(v)$ can be covered by $2^{2\alpha}$ balls of radius $r/4$. Therefore, the number of nodes in $B_r(v)$ is at most $2^{2\alpha}$. □

In combination with the above MIS algorithms for bounded degree graphs, Lemma 8.16 implies the following corollary.

Input: coloring with colors $\{1, \ldots, K\}$
Output: coloring with colors $\{1, \ldots, \Delta + 1\}$
1: **for** $c := 1$ **to** K **do**
2: **send** color(v_i) to all neighbors;
3: **if** color$(v_i) = c$ **then**
4: color(v_i) := minimal possible color
5: **end if**
6: **end for**

Algorithm 8.4: Color Reduction (node v_i)

Corollary 8.17. *The time complexity of a single iteration of the while-loop of Algorithm 8.3 is $O(\log^* n + 2^{4\alpha})$, that is, for constant doubling dimension, the time complexity is $O(\log^* n)$.*

Finally, consider the time complexity of Lines 9 and 10 of Algorithm 8.3. By Lemma 8.15, each node has a cluster leader in its neighborhood. Line 9 can thus be computed in a single communication round. The time complexity of Line 10 is more interesting. Similar to the construction of an MIS, a distributed coloring algorithm which colors a graph with K colors in time t can be turned into a coloring algorithm which colors a graph with $\Delta + 1$ colors in time $t + K$ (Δ is the maximum degree). Algorithm 8.4 shows how this can be achieved.

By Lemma 8.15, the maximum degree of the cluster graph is at most $2^{4\alpha} - 1$. Using the algorithm of [166] for computing the initial coloring, this results in the following corollary.

Corollary 8.18. *The time complexity of Line 10 of Algorithm 8.3 is $O(\log^* n + 2^{8\alpha})$, that is, for constant doubling dimension, the time complexity is $O(\log^* n)$.*

Fast Implementation of the Basic Algorithm

Because each while-loop iteration has a time-complexity of $O(\log^* n)$ and the number of while-loop iterations is $\lfloor \log(1/d_{\min}) \rfloor$, a straightforward distributed implementation of Algorithm 8.3 has time-complexity $O(\log^* n \cdot \log(1/d_{\min}))$ in doubling unit ball graphs. For $d_{\min}$ tending to 0, this is unbounded. In this section, we will have a second look at the complexity of Algorithm 8.3 resulting in significantly better result.

As described in the discussion of the $\mathcal{LOCAL}$ model in Section 4.2, every distributed k-round algorithm can in principle formulated as follows.

1. Each node collects its complete k-neighborhood in k communication rounds

2. Each node computes the output by locally simulating the relevant part of the distributed algorithm (no communication needed)

With this locality-preserving transformation in mind, consider a single iteration of the while-loop of Algorithm 8.3. All communication needed to

compute an iteration of the while-loop is between nodes in $\mathcal{G}_r$. Hence, all messages are sent on edges which have length at most r. If the number of communication rounds in this iteration is k and if all messages of those k rounds are on edges of length at most r, then every node can obtain information from distance at most $k \cdot r$. In other words, every node is able to compute the outcome of the while-loop iteration locally after collecting the complete neighborhood up to distance $k \cdot r$ (w.r.t. the metric). That is, nodes have to collect all information which is accessible by paths of length at most $k \cdot r$. By the triangle inequality, it is possible to collect this information in $2kr$ rounds. Applying this transformation to Algorithm 8.3 leads to the following lemma.

Lemma 8.19. *Algorithm 8.3 can be implemented with a time complexity of $O(\log^* n + 2^{8\alpha})$. For constant doubling dimension, the algorithm requires $O(\log^* n)$ communication rounds.*

Proof. By Corollary 8.17, the number of rounds of an iteration of the while-loop of Algorithm 8.3 is $O(\log^* n + 2^{4\alpha})$. In an iteration with radius r, nodes therefore need to collect information from distance at most $O(r \cdot (\log^* n + 2^{4\alpha}))$. In order to locally compute the result of *all* $\lfloor \log(1/d_{\min}) \rfloor$ iterations, the distances from which information has to be collected must be summed up over all iterations. However because r grows exponentially by a factor of 2 in each iteration, we have a geometric series and can upper bound the sum by taking 2 times the maximum summand. Therefore, every node can compute the outcome of the entire while-loop with information from its $O(\log^* n + 2^{4\alpha})$ neighborhood. In the $\mathcal{LOCAL}$ model, this can therefore be implemented in $O(\log^* n + 2^{4\alpha})$ communication rounds. Together with Corollary 8.18, we get the required result. □

Note that when collecting the whole neighborhood, nodes do not need to know the minimum distance $d_{\min}$. As the radius grows exponentially, the locality of the problem is independent of the starting radius. Specifically, each node can use the minimum distance in the collected neighborhood in order to locally simulate Algorithm 8.3. The complete algorithm to compute a $(O(1), O(1))$-decomposition in doubling UBG's can be summarized as follows.

1. exchange 1-hop distances with neighbors
2. locally compute the while-loop of Algorithm 8.3 beginning from radius $r \in O(1/(\log^* n + 2^{4\alpha}))$ up to the radius for which it suffices to know the 1-neighborhood.
3. collect $O(\log^* n + 2^{4\rho})$-neighborhood (it is sufficient to only collect data about nodes which are still in $\mathcal{V}$)
4. compute the remaining iterations of the while-loop
5. compute clusters and cluster coloring (Lines 9,10 of Algorithm 8.3)

Computing the solution for small radii first and then collecting the rest of the neighborhood is done in order to obtain reasonable message sizes, as shown in the following main theorem.

Theorem 8.20. *In the unit ball graph model, the above algorithm computes a $(2, 2^{4\alpha})$-decomposition in time $O(\log^* n + 2^{8\alpha})$, where α is the doubling dimension of the underlying metric. Given that all distances and node IDs can be represented by K bits, the maximal message size is at most*

$$O\left(\left[\left(\log^* n + 2^{4\alpha}\right)^{O(\alpha)} + \Delta\right] \cdot K\right)$$

bits. Hence, for constant α, the time complexity is $O(\log^* n)$ and largest message requires at most $O(((\log^* n)^{O(1)} + \Delta)K)$ bits.

Proof. The time complexity follows from Lemma 8.19. For the correctness of the algorithm, it remains to prove that only collecting information about nodes in $\mathcal{V}$ for $r \geq O(1/(\log^* n + 2^{4\alpha}))$ (steps 3 and 4) is sufficient. Because all communication of Algorithm 8.3 is on $\mathcal{G}$, this is however clear.

For the bound on the message size, we need to have a closer look at steps 1, 3, and 5 where messages are exchanged. In step 1, all nodes send at most Δ distances and node IDs to their neighbors. This requires messages of size $O(\Delta \cdot K)$. In step 3, a message can contain at most the entire R-neighborhood of a node, where $R := O(\log^* n + 2^{4\alpha})$. Let N be the maximum number of nodes which such a R-neighborhood can contain. If $r \in \Theta(1/(\log^* n + 2^{4\alpha}))$ denotes the largest radius for which the while-loop has been computed in step 2, we know that for all pairs of nodes $u, v \in \mathcal{V}$, we have $d(u, v) > r$. Therefore, every ball of radius $r/2$ contains at most 1 such node. By the definition of α, the maximum number of nodes in a ball of radius R is at most

$$N \;\leq\; (2^{\alpha})^{(\log_2(R/r)+1)} \;=\; \left(\frac{R}{r}\right)^{\alpha+1}.$$

The number of edges in the R-neighborhood is at most quadratic in N. The theorem now follows from the definition of R and r. □

Remark:
The results of this section can be regarded from a more information-theoretic angle as well. Assume, for instance, that we are given a doubling metric (X, d). All points in X have to provide their part of the solution of a global problem. Thereby, each member $x \in X$ has to base its decision on information available in the ball $B_r(x)$ for some radius r. Theorem 8.20 shows that choosing the radius $r \in O(\log^* n)$ suffices for obtaining a solution to many natural problems. As a particular example, we might wish to construct an ϵ-*net*, that is, we want to select a set of points S such that any two selected points have distance at least ϵ and such that any point $x \in X$ has a point in S at distance less than ϵ. In algorithms for metric spaces, ϵ-nets are a widely used structure (e.g. [121]). Theorem 8.20 proves that every node can decide whether it is in the ϵ-net S based on information available in its $O(\epsilon \log^* n)$-neighborhood only.

8.4 Local Approximation Schemes

As mention at the beginning of this chapter, many classical graph theory problems are easy to approximate on UDGs. For instance, an MIS is a constant approximation for MDS and a $(\Delta + 1)$-coloring is a constant approximation for the minimum vertex coloring problem, which is well-known to be notoriously hard in general graphs. Our MIS and decomposition algorithms in Sections 8.2 and 8.3 therefore yield constant approximation algorithms to these problems.

When it comes to centralized, sequential algorithms, however, even better solutions are known for UDGs and more general geometric intersection graphs. Particularly, such graphs allow for *polynomial approximation schemes (PTAS)* for various combinatorial optimization problems, including MDS [127]. Typically, such polynomial approximation schemes exploit the geometric representation of the graphs (i.e., the nodes' coordinates in $\mathbb{R}^2$ in UDGs), and then apply a so-called *shifting strategy* in order to achieve an arbitrarily good approximation ratio [25, 124]. Among the numerous subsequent algorithms employing the shifting algorithms, we would like to point out [82], in which a PTAS for the maximum independent set problem on general disk graphs is described, and [41], which presents a nice PTAS for the *minimum connected dominating set* problem.

Unfortunately, approaches based on the shifting strategy appear to be *inherently central* and cannot efficiently be adapted to work in a distributed context. In this strategy, the graph is partitioned into boxes or stripes, e.g. for independent set construction by removing all vertices alongside designated boundaries. Then, a candidate solution is created by solving each component separately and combining the partial subsets. This is done for several disjunctive and exhaustive boundaries, and the best overall candidate solution is returned as the desired solution. Clearly, such an approach requires some sort of centralized control for gathering the partial solutions and deciding on the best solution among these, and hence, the shifting strategy appears to be unsuited for use in distributed computing.

Even when considering centralized approaches, however, the case where the geometric representation of the underlying graph is not given is significantly different: For unit disk graphs, it is known that given the graph, it is NP-hard to computing its representation [37], or even a good approximation thereof [38, 148]. Nonetheless, it turns out that the topological structure of UDGs allows to devise a PTAS for MDS and related problems even in the absence of the graph's geometric representation. Specifically, Nieberg, Hurink, and Kern present beautiful PTAS' for the maximum independent set problem and MDS on unit disk graphs *without* a representation [185, 186].

Whereas approximation schemes for geometric intersection graphs have thus been studied extensively in the centralized case, no similar results are known in *distributed settings*, where each node has to base its decision on information gathered in its local neighborhood via communication. In this section, we address this by extending the work of [185, 186] to obtain the first *distributed approximation schemes* for the minimum dominating set and the maximum independent set problems in graphs with bounded independence.

Specifically, we present algorithms for the two problems that compute a $(1+\varepsilon)$-approximate solution in time $O(T_{\text{MIS}} + \log^* n/\varepsilon^{O(1)})$, where $\varepsilon > 0$ can be chosen arbitrarily and T_{MIS} denotes the time for computing an MIS in the network graph.

Locally Optimal Subsets

We begin by showing how a sequential algorithm can compute a maximum independent set or a minimum dominating set in the bounded neighborhood of a node, such that the local solution meets the desired approximation ratio of $1+\varepsilon, \varepsilon > 0$.

Let $v_0 \in V$ be an arbitrary node in a f-bounded independence graph G, where $f(r)$ is polynomial in r. For the maximum independent set problem, consider v_0's r-neighborhood $\Gamma_r(v_0)$ for increasing radii $r = 0, 1, 2, \ldots$, and compute a maximum independent set $I_r \subset \Gamma_r(v_0)$ as long as the condition

$$|I_{r+1}| > (1+\varepsilon)|I_r|$$

holds. Let $\hat{r}$ denote the smallest radius r for which the above criterion is violated. For $\hat{r}$, the following claim holds (an analogous lemma for unit disk graphs was proven in [186]).

Lemma 8.21. ***(Maximum Independent Set)*** *Let $G = (V, E)$ be a graph of polynomially bounded independence. There exists a constant $c = c(\varepsilon)$ such that $\hat{r} \leq c$.*

Proof. . Due to the structure of the graph G, we have $|I_r| \leq f(r)$. From the definition of $\hat{r}$, it holds that for every $r < \hat{r}$ the following inequalities hold

$$f(r) \geq |I_r| > (1+\varepsilon)|I_{r-1}| > \ldots > (1+\varepsilon)^r |I_0| = (1+\varepsilon)^r.$$

Because $(1+\varepsilon)^r$ exceeds any polynomial in r for a large enough constant r, the claim follows. □

The above lemma implies that the radius of the largest neighborhood of v from which information is required by v_0 is also bounded by a constant $\hat{r}$ that only depends on the desired approximation ratio. Furthermore, the computations to be performed locally by v_0 are bounded by $n^{O(f(c)^2)}$.

For the MDS problem, vertex v_0 computes for $r = 0, 1, 2, \ldots$, a minimum cardinality set D_r that dominates $\Gamma_r(v_0)$ as long as

$$|D_{r+2}| > (1+\varepsilon)|D_r|$$

holds. Again, denote by $\hat{r}$ the smallest r which violates the above.

Lemma 8.22. ***(Minimum Dominating Set)*** *Let $G = (V, E)$ be a graph of polynomially bounded independence. There exists a constant $c = c(\varepsilon)$ such that $\hat{r} \leq c$.*

Proof. It follows from a similar argument as in Lemma 8.21 that for even $r < \hat{r}$, it is

$$f(r) \geq |I_r| \geq |D_r| > (1+\varepsilon)|D_{r-2}| > \ldots > (1+\varepsilon)^{\frac{r}{2}}|D_0| = (\sqrt{1+\varepsilon})^r.$$

For uneven r, the same chain of inequalities holds, and the claim follows. □

Given Lemmas 8.21 and 8.22, a centralized PTAS for the maximum independent set problem and for MDS can be obtained as follows (for the UDG case, see [185, 186]): Select an arbitrary vertex v_0 and compute a subset of its bounded neighborhood $\Gamma_{\hat{r}}(v_0)$ that meets the desired criterion. In the maximum independent set case, compute a locally optimal independent set $I_{\hat{r}}(v_0)$ for this neighborhood. When the expansion stops, remove the neighborhood $\Gamma_{\hat{r}+1}(v_0)$ from G, and combine $I_{\hat{r}}$ with the partial solution obtained thus far. Then, move on the next iteration with the remaining vertices of G. Because in each iteration, the neighborhood removed is larger than the neighborhood in which independent set nodes are selected, the resulting solution indeed forms an independent set.

For the MDS case, we must account for the possibility of nodes outside a subset being capable of dominating nodes inside this subset. Therefore, locally optimal dominating sets are always created with respect to G. For neighborhoods $\Gamma_r(v_0)$, it holds that $D_r \subseteq \Gamma_{r+1}(v_0)$. Therefore, we remove in every iteration $\Gamma_{\hat{r}+2}(v_0)$ from G, and add $D_{\hat{r}+2}$ to the partial solution, before again going on with the remaining graph. Again, the reason for considering $(\hat{r}+2)$-neighborhoods is that $D_r \subset \Gamma_r(v_0)$ may not hold, but $D_r \subset \Gamma_{r+1}(v_0)$ must be satisfied.

8.4.1 Local Approximation Schemes

A straightforward way of turning the above PTAS into a distributed algorithm is to distribute the greedy strategy for picking a new central vertex v_0. The problem is, however, that at any point in time, there may only be a single point of activity in the graph, which yields (at least) a linear number of rounds. In this section, we present fast distributed algorithms for this problem.

The distributed approximation schemes are based on the idea of quickly identifying nodes which are separated far enough so that their neighborhoods do not overlap during the construction of the partial solutions. These nodes then construct locally optimal solutions in parallel. In order to select these initial node-set, we compute an MIS in the graph.

Maximum Independent Set

Algorithm 8.5 presents the details of the maximum independent set algorithm. From Lemma 8.21, we know that each partial solution $I_{\hat{r}}(v_0)$ is inside a bounded neighborhood around the central node $v_0 \in V$ if G is polynomially independence bounded. The radius c of the largest neighborhood is bounded by a constant that only depends on the desired approximation guarantee ϵ,

Input: Graph $G = (V, E)$ of polynomially bounded independence
$\varepsilon > 0$, $c := c(\varepsilon) + 2$ (Lemma 8.21)
Output: $(1 + \varepsilon)$-approximate maximum independent set I

```
1: Compute MIS S of G;
2: Construct auxiliary graph G = (S, E_{2c+1});
3: Color G using Δ_G + 1 colors;
4: I := ∅;
5: for k = 1 to Δ_G + 1 do
6:    for every v ∈ S with color k do
7:       while Γ(v) ∩ V ≠ ∅ do
8:          For some u ∈ Γ(v) ∩ V, compute maximum independent set I_r̂ for
            neighborhood Γ_r̂(u) ∩ V such that |I_{r̂+1}| ≤ (1 + ε)|I_r̂|;
9:          Inform vertices in Γ_c(v) about r̂ and I_r̂;
10:         I := I ∪ I_r̂(u);
11:         V := V \ Γ_{r̂+1}(u);
12:      end while;
13:   end for;
14: end for;
```

Algorithm 8.5: Local approximation scheme (Maximum Independent Set)

which is assumed to be known by all nodes. In the maximum independent set approximation scheme, two partial solutions $I_{\hat{r}}(v_1)$ and $I_{\hat{r}}(v_2)$ do not interfere if $d_G(v_1, v_2) \geq \hat{r}_1 + \hat{r}_2 + 1$ and hence, if $d_G(v_1, v_2) \geq 2c + 1$. Non-interfering nodes can add partial solutions to a global solution in parallel.

The sequence in which MIS nodes add partial solutions to the global solution is determined by a coloring of these nodes. In particular, these nodes are colored such that two nodes of the same color do not interfere. Technically, the coloring is done using an auxiliary graph $\mathcal{G} = (S, \mathcal{E}_{2c+1})$ in which two nodes are connected if their distance in the original graph is at most $2c + 1$. Note that the maximum degree $\Delta_{\mathcal{G}}$ of $\mathcal{G}$ is bounded by $f(2c + 1) \in O(1)$ and that for every edge in $\mathcal{E}_{2c+1}$, the two endpoints can communicate in the original graph in a constant number of communication rounds.

There is one technicality that needs to be considered. So far, we have considered only MIS nodes as possible centers of the partial solutions. For each node in G, however, it needs to be ensure that it is considered during the course of the algorithm when not participating in a partial solution of another independent node. By the maximality of the MIS, every node is adjacent to a node in S. In Line 7, if needed, we therefore not only consider nodes in the MIS, but also nodes in the entire neighborhood $\Gamma(v)$ of an MIS node $v \in S$.

Each execution of the while-loop eliminates some vertices from V. Furthermore, all vertices and thus the entire graph are considered in the algorithm by the following lemma.

Lemma 8.23. *Upon completion of Algorithm 8.5, it holds that $V = \emptyset$ and the set I is independent.*

Proof. As for termination, consider the inner while-loop (Line 7) and suppose that a vertex $v \in V$ has not been eliminated. Then, v cannot be adjacent to a vertex in S, a contradiction to the independent set's maximality. For the independence of I, we look at Lines 10 and 11 of the algorithm. While keeping $I_{\hat{r}}(u) \subset \Gamma_{\hat{r}}(u)$ as partial solution, all nodes from $\Gamma_{\hat{r}+1}(u)$ are removed from V. Since MIS nodes of different colors are not considered at the same time, and because two MIS nodes with the same color have a distance of at least $2\tilde{c}+1$ the independence of I follows by the definition of $\mathcal{G}$. □

It remains to show that the independent set I satisfies the desired approximation ratio of $1+\varepsilon$.

Lemma 8.24. *Let I_{OPT} denote an optimal solution to the Maximum Independent Set problem on G. Then, the solution I created by Algorithm 8.5 satisfies*

$$(1+\varepsilon)|I| \;\geq\; |I_{OPT}|.$$

Proof. As already discussed, the neighborhoods $\Gamma_{\hat{r}}(u)$, are either separated due to the coloring of the auxiliary graph $\mathcal{G}$ or are constructed in sequence when considering different colors or two vertices $u, u' \in \Gamma(v)$ for a fixed $v \in S$. Also, every vertex in G is considered in the creation of the neighborhoods. Let $\tilde{\Gamma}_{\hat{r}_u+1}(u)$ denote the respective neighborhoods $\Gamma_{\hat{u}+1} \cap V$, where V is considered with respect to previous steps of the algorithm, and $u \in V$ are the central vertices as chosen in Line 8. These adjusted sets $\tilde{\Gamma}_{\hat{r}_u+1}(u)$ form a partition of G. Furthermore, the partition together with the criterion for the creation of the neighborhoods, i.e., $|I_{\hat{r}+1}| \leq (1+\varepsilon)|I_{\hat{r}}|$, yields

$$\begin{aligned}
|I_{OPT}| &= \Big|\bigcup\nolimits_u \Big(I_{OPT} \cap \tilde{\Gamma}_{\hat{r}_u+1}(u)\Big)\Big| \;=\; \sum\nolimits_u \Big|I_{OPT} \cap \tilde{\Gamma}_{\hat{r}_u+1}(u)\Big| \\
&\leq \sum\nolimits_u |I_{\hat{r}_u+1}(u)| \;\leq\; \sum\nolimits_u (1+\varepsilon)|I_{\hat{r}_u}(u)| \\
&= (1+\varepsilon)\Big|\bigcup\nolimits_u I_{\hat{r}_u}(u)\Big| \;=\; (1+\varepsilon)|I|.
\end{aligned}$$

Note that the summation is over all central vertices u as chosen by the algorithm. □

What is the time complexity of Algorithm 8.5? Let T_{MIS} denote the time for computing a maximal independent set in the pre-processing phase. The coloring of the $\mathcal{G}$, due to its bounded degree, takes $O(\log^* n)$ rounds [58]. The maximum degree of $\mathcal{G}$ depends on the constant $c = c(\varepsilon) + 2$. Since G has polynomially bounded independence, it holds that $\Delta_{\mathcal{G}} \in O(1/\varepsilon^{O(1)})$, where the exponent of $1/\varepsilon$ depends on the polynomial bound of the graph G itself.

As for the second part of the algorithm, Line 6 can be done completely in parallel by the respective MIS nodes of the same colors. For the local neighborhoods of the cluster leaders, we have the following lemma.

Lemma 8.25. *Consider Line 7 of Algorithm 8.5. The while-loop is executed at most $f(1) = O(1)$ times for any $v \in S$.*

Input: Graph $G = (V, E)$ of polynomially bounded independence
$\varepsilon > 0$, $c := c(\varepsilon) + 2$ (Lemma 8.22)
Output: $(1 + \varepsilon)$-approximate Min. Dominating Set D

```
1: Construct MIS S and colored auxiliary graph G = (S, E_{2c+7});
2: D := ∅;
3: for k = 1 to Δ_G + 1 do
4:    for every v ∈ S with color k do
5:       if Γ(v) ∩ V ≠ ∅ then
6:          For some u ∈ Γ(v) ∩ V, compute minimum dominating set D_r̂ of
            Γ_r̂(u) ∩ V such that |D_{r̂+2}| ≤ (1 + ε)|D_r̂|;
7:          Inform Γ_c(v) about r̂ and D_{r̂+2};
8:          D := D ∪ D_{r̂+2}(u);
9:          V := V \ Γ_{r̂+2}(u);
10:      end if;
11:   end for;
12: end for;
```

Algorithm 8.6: Local approximation scheme (Minimum Dominating Set)

Proof. In $\Gamma(v)$, there are at most $f(1)$ independent vertices, and in each round, at least the first order neighborhood of the vertex $u \in \Gamma(v) \cap V$ is eliminated from V. □

Overall, we can therefore bound the number of communication rounds for the second part of the algorithm by $O(c \cdot f(1) \cdot \Delta_{\mathcal{G}}) = O(1/\varepsilon^{O(1)})$ for $\varepsilon > 0$.

Minimum Dominating Set

Devising a similar approximation scheme for the MDS problem now requires only minimal adjustments (Algorithm 8.6). A locally optimal dominating set is computed by the MIS nodes in S. This is done in such a way that the combination of these local subsets dominate the whole graph G, while also obeying the desired approximation ratio. The main difference is that the auxiliary graph is defined with an edge-set $\mathcal{E}_{2c+7}$ in order to maintain enough separation between partial solutions.

In analogy to Lemma 8.23 of the maximum independent set case, we can prove the following lemma.

Lemma 8.26. *Algorithm 8.6 terminates, and at the end, we have $V = \emptyset$. Furthermore, the solution D dominates the graph G.*

Proof. The proof is analogous to the proof of Lemma 8.23. Specifically, the local set $D_{\hat{r}+2}(u)$ dominates $\Gamma_{\hat{r}+2}(u) \cap V$ which is eliminated from V at every step. □

As for the cardinality of D created by the algorithm, the following lemma establishes its approximation ratio.

Lemma 8.27. *Let D_{OPT} denote an optimal solution to the Minimum Dominating Set problem on G. Then, the set $D \subset V$ computed by Algorithm 8.6 satisfies*

$$(1+\varepsilon)|D_{OPT}| \;\geq\; |D|.$$

Proof. Again, let $\tilde{\Gamma}_{\hat{r}_u}(u)$ denote the neighborhoods with respect to V during the execution of the algorithm. By induction over $u \in \Gamma(v), v \in S$, it can be seen that the sets $\tilde{\Gamma}_{\hat{r}_u+1}(u)$ are mutually disjoint. Note in this context that the sets $\tilde{\Gamma}_{\hat{r}_u}(u)$ created in parallel (Line 4) are non-overlapping and can thus w.l.o.g. be considered in arbitrary order for central vertices of the same color. Furthermore, $\tilde{\Gamma}_{\hat{r}_u+1}(u) \cap D_{OPT}$ dominates $\tilde{\Gamma}_{\hat{r}_u}(u)$ in G. Hence,

$$\begin{aligned}
|D_{OPT}| \;&\geq\; \Big|\bigcup\nolimits_u \Big(D_{OPT} \cap \tilde{\Gamma}_{\hat{r}_u+1}(u)\Big)\Big| \;=\; \sum\nolimits_u \Big|D_{OPT} \cap \tilde{\Gamma}_{\hat{r}_u+1}(u)\Big| \\
&\geq\; \sum\nolimits_u |D_{\hat{r}_u}(u)| \;\geq\; \sum\nolimits_u \frac{1}{1+\varepsilon}|D_{\hat{r}_u+2}(u)| \\
&\geq\; \frac{1}{1+\varepsilon}\Big|\bigcup\nolimits_u D_{\hat{r}_u+2}(u)\Big| \;=\; \frac{|D|}{1+\varepsilon}.
\end{aligned}$$

□

The number of rounds needed in the second part of Algorithm 8.6 is $O(1)$ for fixed $\varepsilon > 0$ by the same argument as in the maximum independent set case. More precisely, we can summarize the results in the following theorem.

Theorem 8.28. *Let $G = (V, E)$ be a polynomially independence bounded graph. For any $\varepsilon > 0$, there exist local, distributed $(1+\varepsilon)$-approximation algorithms, for the Maximum Independent Set and Minimum Dominating Set problems on G. The number of communication rounds needed for the respective construction of the subsets is $O(T_{MIS} + \log^* n/\varepsilon^{O(1)})$.*

When combining this result with Theorems 8.13 and 8.20 of Sections 8.2 and 8.3, we obtain the following corollaries.

Corollary 8.29. *Let G be a graph with polynomially bounded independence. For any constant $\varepsilon > 0$, a $(1+\varepsilon)$-approximation for MDS and the maximum independent set problem can be computed deterministically in time $O(\log \Delta \log^* n)$ in the $\mathcal{LOCAL}$ model.*

Corollary 8.30. *Let G be a doubling unit ball graph. If nodes know the distances to their neighbors, for any constant $\varepsilon > 0$, a $(1+\varepsilon)$-approximation for MDS and the maximum independent set problem can be computed deterministically in time $O(\log^* n)$ in the $\mathcal{LOCAL}$ model.*

Chapter 9

Conclusions and Outlook

In view of ever growing distributed systems and global networks, it is becoming increasingly vital to understand distributed algorithms in which the participating entities do not require full information about the state of the network. In many large-scale networks, routing and infrastructure maintenance tasks such as clustering or scheduling must instead be handled *locally*. Furthermore, fast local algorithms are also desirable when coping with aspects such as dynamics, mobility, or churn, which are key characteristics of peer-to-peer networks or mobile ad hoc networks. With this in mind, it can be argued that in large-scale distributed systems, all efficient computation is inherently local, which highlights the need to understanding the possibilities and limitations of *locality-sensitive approaches* [196] and *local computation*.

Besides its practical importance in networking, the study of local computation also poses fascinating *theoretical* questions. As we have seen in this part of the thesis, studying the locality of optimization problems sheds new light into their combinatorial structure. The field of local computation is thus strongly related to well-established areas of theoretical computing science, including approximation theory, distributed computing, geometry, or graph theory and discrete mathematics in general.

A particularly interesting relation exists between local computation and the (economic) "value of information". As we have seen, the $\mathcal{LOCAL}$ model yields an equivalence between the time complexity of a distributed algorithm and the amount of *local information* on which distributed decision makers must base their decision. Local computation therefore corresponds to a particularly important special case of *distributed decision-making with incomplete information* as studied in [192, 193]. Our upper and lower bounds on the amount of locality required to achieve a certain quality of a global optimization in Chapters 6 and 7 almost tightly capture the *value of information* of each additional increase if the neighborhood. Particularly our lower bounds describe the degradation of distributed decisions caused by lack of information between cooperating agents.

Quantifying the suboptimality of a global optimum due to lack of global

knowledge can also be regarded as studying the *"price of locality"*. The notions of *approximation ratio* (approximation algorithms) and *competitive ratio* (online algorithms) have long been used as a measure to quantify the loss of efficiency due to lacking computational power and knowledge about the future, respectively. More recently, researchers have investigated the so-called *"price of anarchy,"* or coordination ratio, which captures the degree of suboptimality resulting from *selfish behavior* [190, 208]. In a similar spirit, our work on local computation sheds light on the *"price of locality,"* i.e., the degradation of a globally optimal solution if each individual's knowledge is restricted to its neighborhood or local environment. While approximation and online algorithms have been thoroughly studied, and the price of anarchy has recently received a lot of attention, much less is known about this price of locality in comparison.

In this context, it is clear that deriving *lower bounds* on local computation is of particular interest. So far, Linial's $\Omega(\log^* n)$ lower bound for computing an MIS on a ring [166] has been standing as the major lower bound on locality that resulted from locality restrictions only. With our lower bounds of Chapter 7, we hope to have contributed to a better understanding of the "price of locality".

There are numerous directions for future research and there exists a bulk of very recent work that deals with local computation for problems such as MST [75, 76], spanners [66, 69], or coloring [152]. It is interesting to study the locality nature of other network coordination problems that appear to have some kind of local structure. One such problem is, for instance, the *maximum domatic partition problem* [88]. In this problem, the goal is to select as many disjoint dominating sets as possible. A simple adaptation of the randomized algorithm in [88] shows that in a constant number of communication rounds, an $O(\log n)$ approximation to the problem can be found. In this sense, the maximum domatic partition problem is more local than, say, the maximum matching or the MVC problem. Faster distributed algorithms for the domatic partition problem in graphs of bounded independence and doubling unit ball graphs have recently been presented in [198]. Nonetheless, the exact time-approximation trade-off of this problem is still to be revealed. Other interesting local problems may include the *maximum unique coverage problem* [68], various variants of facility location problems, and–of course–coloring problems.

Beyond these specific open problems, the most intriguing *distant goal* of this line of research is to divide distributed problems into *complexity classes* according to the problems' local nature. The existence of locality-preserving reductions and the fact that several optimization problems exhibit similar characteristics with regard to locality raises the hope for something like a *locality hierarchy* of distributed problems. It would be particularly interesting to establish ties between this distributed hierarchy of complexity classes and the classic complexity classes originating in the Turing model of computation.

Besides classifying computational problems, studying local computation may also help in gaining a more profound understanding of the relative strengths of the underlying network models themselves. As we have seen in Chapter 8, for instance, the vast family of unit ball graphs is—from a

distributed complexity point of view—equally hard as a simple ring network. In both settings, an MIS or a decomposition can be computed in asymptotically optimal time $\Theta(\log^* n)$. On the other hand, our lower bounds prove that general graphs are strictly harder, thus separating these computational models with regard to their distributed complexity and locality.

The notion *distributed approximation* is of interest even beyond locality in graphs. As already pointed out in our discussion of the $\mathcal{CONGEST}$ model, for instance, many problems are hard to approximate in a distributed way even in complete graphs or graphs with small diameter if message size is limited. While there are some beautiful results in this model (most notably the $O(\log \log n)$-time MST algorithm of [168]), there lacks a systematic understanding of the limiting factors in such very low-diameter graphs. While locality or information-theoretic restrictions do not appear to be the decisive bottlenecks in complete graphs, other aspects such as "coordination" become increasingly important. Formally capturing these intuitive notions and devising corresponding lower bounds for such networks appears to be a challenging task for the future. Notice that studying low-diameter (or even complete) graphs is interesting from a networking point of view, because these graphs can be regarded as abstracting the network on top of an overlay-network that provides point-to-point connections between all pairs of nodes.

Finally, locality is only one characteristic of modern large-scale networks. Another important aspect of many modern distributed systems is that the participating entities are often selfish agents [190]. Part IV of this thesis studies the impact of selfishness in networks in more detail and it would be intriguing to explore connections between *locality* and *selfishness*.

Part II

Radio Networks

Chapter 10

Wireless Ad Hoc and Sensor Networks

The advent of wireless communication in virtually all aspects of daily life has not only spawned a multi-billion dollar market, but has also changed people's lifestyle. While heterogenous *cellular networks* have become omnipresent, a novel, complementing network paradigm has been rapidly emerging in recent years: *Ad hoc and sensor networks*. These networks are a manifestation of the continuing miniaturization of electronics in general and wireless communication technology in particular. Ad hoc and sensor networks are formed of autonomous (sometimes mobile) devices consisting of, among other components, a processor, some memory, a radio communication unit, and a power source, typically a battery. Sensor networks can be considered a specialization of ad hoc networks in which nodes are equipped with sensors measuring certain physical values (humidity, vibration, acceleration, temperature, brightness, etc...).

Ad hoc and sensor networks have been envisioned in various application fields and there is a growing number of real (even commercial) systems being built, ranging from environmental monitoring, surveillance, to medical applications, the observation of chemical and biological processes, disaster relief, and community mesh networks. Besides these application scenarios, it must also be mentioned that—inevitably—ad hoc and sensor network technology also has the potential of finding military applications.

The characteristics of the often highly constrained ad hoc or sensor nodes render the design of efficient and robust protocols for ad hoc and sensor networks particularly challenging. It is therefore not surprising that in the recent networking literature, a plethora of protocols have been proposed for these networks. Often, these new protocols have been of heuristic nature, typically evaluated by means of simulation. From an algorithmic point of view, however, there remain numerous important and interesting unresolved questions. Principally, many of the well-known algorithmic paradigms and results

from traditional networking and distributed computing carry over directly to wireless ad hoc and sensor networks. The specific wireless, infrastructure-less, resource-limited, and potentially mobile nature of these networks, however, also bring about algorithmic challenges that require novel solutions. This is particularly the case when studying link layer or MAC layer issues in these networks.

As mentioned above, one of the most challenging peculiarities of ad hoc and sensor networks is their inherent lack of built-in a-priori infrastructure. Typically, if two nodes are not within mutual transmission range, they can communicate through intermediate nodes relaying their message. In other words, the communication infrastructure must be provided and maintained by the nodes themselves. During and shortly after the deployment of an ad hoc or sensor network, there is no established pattern based on which nodes could efficiently communicate. There is no reliable point-to-point abstraction to higher-layer protocols and applications, messages may collide, and nodes may not even be aware of their neighbors. Before the network can actually start carrying out its intended task, nodes must establish an initial structure or MAC layer, based on which more sophisticated protocols or applications can be employed.

The quintessential task after a network's deployment is therefore the self-organized transition from an unstructured to a structured multi-hop radio network, i.e., the network's *initialization*. In practice, the initialization problem is important and current initialization procedures tend to be slow. Even in a single-hop ad hoc network such as Bluetooth and for a small number of devices, the time-consumption for establishing a reasonable communication pattern is considerable. Clearly, the situation is even worse in a multi-hop scenario with a large number of nodes.

Distributed algorithms for *coloring* and *clustering* appear to be tailor-made for the purpose of bringing structure into a chaotic network and to facilitate the effects of lacking a-priori infrastructure and missing knowledge. The difficulty in applying these algorithms, however, is that any protocol that aims at setting up an initial infrastructure must not rely on any previously established structure. In this sense and in this setting, coloring and clustering algorithms that have been studied in networking and distributed computing typically suffer from one of the two shortcomings: Either the protocols are of heuristic nature and evaluated by simulations only, or, when it comes to algorithms with provable worst-case guarantees, the underlying communication model abstracts away many of the harsh realities that crucially determine the complexity of the task at hand.

Previously, the efficiency of distributed algorithms for computing structures (such as dominating sets, MIS, colorings, ...) that could serve as an initial infrastructure has often been studied in message passing models, or under other strong model assumptions. For instance, it is frequently (and sometimes implicitly) assumed that all nodes wake up and start the algorithm at the same time or that every node knows its neighbors or 2-hop neighbors at the outset. These and other often adopted model assumptions are typically denounced by practitioners as unrealistic and oversimplifying. Particularly during the deployment phase, it is the *absence of coordination,*

infrastructure, and knowledge that ultimately determines the complexity of network coordination tasks. When abstracting away these aspects by assuming too much a-priori knowledge or coordination, the obtained solutions have rarely been directly applicable in practical settings.

Studying coloring and clustering algorithms in the message passing model implicitly implies the existence of an established, underlying medium access control (MAC) layer protocol that provides reliable point-to-point connections to higher-layer protocols and applications. Studying network coordination problems such as clustering or coloring in the absence of an established MAC layer highlights the following *chicken-and-egg* problem: A MAC layer ("chicken") helps achieving a clustering/coloring ("egg"), and vice versa. The problem is that in a newly deployed ad-hoc or sensor network, there is no structure, i.e. there are neither "chickens" nor "eggs."

In this part of the thesis, we aim at bridging or narrowing this gap between theory and practice. Based on a novel *unstructured radio network model*—which is an adaptation of the classic standard radio network model—we evaluate the complexity of clustering and coloring tasks in newly deployed unstructured networks. On the one hand, our model attempts to closely capture the harsh conditions of real ad hoc and sensor systems. On the other hand, we want the model to remain appealing to the *theory of distributed computing*, i.e., the model should be concise enough to allow for stringent proofs and reasoning, preventing simulations from being the only resort to evaluate the algorithms' performance.

The sequel of this part of the thesis is organized as follows. Chapter 11 introduces the *unstructured radio network model*, which captures many of the characteristics of wireless ad hoc and sensor networks, particularly during and immediately after their deployment. Chapters 12 explores the distributed complexity of coloring in unstructured radio networks. In subsequent Chapter 13, we extend these techniques in order to compute an MIS in time $O(\log^2 n)$ with high probability, which is asymptotically optimal given a lower bound for clear radio transmission in [86]. Having established the complexity of network coordination structures in the unstructured radio network model, we present an application of these initial structures in Chapter 14. Specifically, in certain settings, employing initial structures can help in improving the energy-latency trade-off during the deployment phase of wireless sensor networks.

Chapter 11

Unstructured Radio Network Model

In many ways, familiar distributed computing communication models such as the *message passing model* do not describe the harsh conditions faced in wireless ad hoc and sensor networks closely enough, rendering results obtained in these models bear little importance in practice. Ad hoc and sensor networks are multi-hop radio networks and hence, messages being transmitted may interfere with concurrent transmissions leading to collisions and packet losses. Furthermore, all nodes sharing the same wireless communication medium leads to an inherent broadcast nature of communication. A message sent by a node can be received by all nodes in its transmission range. These aspects of communication are modeled by the *radio network model*.

Based on the classic radio network model, we seek to model the following characteristics.

- Ad hoc and sensor networks are *multi-hop*, that is, there exist nodes that are not within their mutual transmission range. Therefore, it may occur that some neighbors of a sending node receive a message, while others experience interference from other senders and do not receive the message.

- Nodes can wake up *asynchronously* at any time. In a wireless, multi-hop environment, it is realistic to assume that some nodes wake up (e.g. become deployed, or switched on) later than others. Because nodes do not have access to a global clock, we do not assume local clocks to be synchronized. Notice that in contrast to work on the *wake-up problem in radio network* [104], nodes are *not woken up* by incoming messages. That is, before its wake-up, a node does not participate in the network in any way; it neither transmits nor receives any messages. See Section 11.2 for more details on the wake-up problem in radio networks.

- Nodes do not necessarily feature a reliable *collision detection* mechanism. This assumption is often realistic, considering that nodes may be tiny sensors with equipment restricted to the minimum due to limitations in energy consumption, weight, or cost. Moreover, the sending node itself does not have a collision detection mechanism either. Hence, a sender does not know how many (if any at all!) neighbors have received its transmission correctly.

- At the time of their waking-up, nodes have only limited knowledge about the total number of nodes in the network and no knowledge about the nodes' distribution or wake-up pattern. Particularly, they have no a-priori information about the number of neighbors and when waking up, they do not know how many neighbors have already started executing the algorithm.

All these restrictions suggest that we deal with a particularly harsh model of computation and naturally, algorithms for such uninitialized, chaotic networks have a different flavor compared to "traditional" algorithms that operate on a given network graph that is static and well-known (at least locally) to all nodes.

11.1 Model and Notation

In the *unstructured radio network model*[1], we consider *multi-hop radio networks without collision detection*. In this model, time is considered to be divided into time slots and in every time slot, a node can either transmit or not transmit a message. A node is able to correctly receive a message only if *exactly one* of its neighbors transmitted in this time slot. If more than one neighbor transmitted, a *collision* occurs and the message becomes garbled. The absence of a collision detection mechanism means that nodes are unable to distinguish between the situation in which two or more neighbors are sending and the situation in which no neighbor is sending. For the sake of simplicity, we assume time to be divided into discrete time slots that are synchronized between all nodes. As long as the nodes' internal clocks run at the same speed, this idealistic assumption incurs only a constant-factor overhead [222].

In contrast to previous radio network models, the unstructured radio network model allows nodes to wake up *asynchronously* at any time, i.e., an adversary may select the wake-up time of each node in a worst-case manner. All nodes are initially assumed to be asleep (i.e., not yet deployed or switched on). Once a node v is activated, it starts executing its protocol $\mathcal{Q}$, which among other things dictates in which time slots v transmits a message. Thereby, the protocol $\mathcal{Q}$ can depend only on the *local* clock of v (the number of time slots since its activation), and possibly additional information received earlier from other nodes. In other words, there is *no globally*

[1] In a subsequent paper, an essentially equivalent model was dubbed *weak sensor model* [85].

synchronized clock and upon waking up, a node has no information as to whether it is the first to wake up, or whether other nodes have been running the algorithm for a long time already. As for terminology, we call a node *sleeping* or *asleep* before its wake-up, and *awake* thereafter.

The primary complexity measure we will consider is the time complexity defined in the following.

Definition 11.1 (Time Complexity). *The* running time T_v *of a node* v *is defined as the number of time slots between* v*'s* waking up *and the time* v *makes an irrevocable* final decision *on the outcome of its protocol. The* time complexity $T(\mathcal{Q})$ *of algorithm* $\mathcal{Q}$ *is defined as the maximum running time over all nodes in the network, i.e.,* $T(\mathcal{Q}) := \max_{v \in V} T_v$.

We model the network as an undirected graph $G = (V, E)$, where two nodes u and v can communicate with each other if there is an edge $(u, v) \in E$. In order to capture the wireless characteristics of ad hoc and sensor networks, we again assume the *bounded independence model* (see Chapter 8) in which there can be at most $\kappa_1 = f(1)$ mutually independent nodes in the 1-hop neighborhood of any node. Similarly, there are at most $\kappa_2 = f(2)$ nodes in the 2-hop neighborhood of any node. Note that due to asynchronous wake-up, some nodes may still be asleep, while others are already transmitting. Hence, at any time, there may be sleeping nodes which do *not receive* a message in spite of there being a communication link between the two nodes. In other words, nodes simply do not exist in the network until the time of their wake-up.

When waking up, nodes have only scarce knowledge about the network graph's topology. In particular, a node has no information on the number of nodes in its neighborhood. However, every node has estimates n and Δ for the number of nodes in the network and the maximum degree, respectively. In reality, it may not be possible to foresee these global parameters precisely by the time of the deployment, but it is usually possible to pre-estimate rough bounds.

Every node has a unique identifier, but identifiers may not be in the range $1, \ldots, n$. Particularly, none of our algorithms performs explicit operations on the node's identifiers. Instead, IDs are merely used to let a receiver recognize whether two messages were sent by the same sender.[2] Finally, like in the $\mathcal{CONGEST}$ model of Section 4.3, message size is restricted to $O(\log n)$ bits, assuming that the identifier space is polynomial in n. That is, every message may contain at most a constant number of node identifiers.

11.2 Related Work

The first communication network with the semantics of a (single-hop) radio network appears to be the Alohanet system developed at the university of

[2] In some papers on wireless sensor networks, it is argued that sensor nodes do not feature any kind of unique identification (such as a MAC number, for instance). Clearly, in such a case, each node can obtain a unique ID by simply choosing a random number from a large enough range, say $[1 \ldots n^3]$, upon waking up.

Hawaii in the 1970's. This system brought about the well-known *Aloha protocol* for regulating access to the channel. Another famous conflict-resolution protocol for a multiple access channel is the *Ethernet* protocol, whose conflict-resolution scheme uses exponential backoff. Considering the importance of these and other multiple access channels, it is not surprising that there exists a substantial work that analyzes capacity, throughput, and stability of different protocols in a variety of models. For a thorough algorithmic survey of these channel access protocols and their analysis, we refer to [47].

More recently, additional algorithmic problems have been studied in single-hop radio networks including the so-called *initialization problem*—assigning unique identities from 1 to n to a set of n identical anonymous nodes—[182], approximating the number of participating stations n [137], or leader election [183].

The model of *multi-hop radio networks* was introduced by Chlamtac and Kutten [45] who considered sequential algorithms to find an efficient broadcast protocol for a given input network. Ever since, research on radio networks has flourished, rendering the model one of the most important models in distributed computing. The first efficient distributed (probabilistic) broadcast protocol was presented in [30]. The broadcasting problem has remained the most widely studied subject in the literature on radio networks and there have been a number of efficient probabilistic and deterministic solutions to the problem, e.g. [46, 48, 63]. Alon, Bar-Noy, Linial, and Peleg show that, even for graphs with diameter 2, $\Omega(\log^2 n)$ rounds may be necessary [10]. The other well-known lower bound was given by Kushilevitz and Mansour, showing that for every (possibly randomized) broadcast algorithm, there exists a network of n nodes and diameter D, such that the expected broadcasting time of the algorithm is $\Omega(D \log(n/D))$ [155]. Besides the broadcasting problem, there has also been extensive work on other communication primitives such as all-to-all communication (called *gossiping*, e.g. [50]). Furthermore, there have been results on the feasability of broadcast under the presence of Byzantine or faulty processors in the network [142]

The above papers are typically based on the simplifying assumption that all nodes wake up at the same time, i.e., every node is awake and ready to receive a message at the outset of (and throughout) the algorithm. Note that this implies the existence of a *globally synchronous clock* to which all nodes have access. In many settings—especially in wireless or dynamic ones—this assumption may be too idealistic. An important shift towards studying settings with *asynchronous wake-up* has been initiated by Gąsieniec, Pelc, and Peleg with their studying the so-called *wake-up problem* in radio networks [104]. In this problem, all nodes are initially assumed to be asleep. Each node v in the network can either wake up spontaneously, or can be activated by receiving a message from one of its neighbors. That is, in this setting, messages act as wake-up signals. Once a node is activated, it starts executing its wake-up protocol Q, which determines in which time slots v transmits a message. In this model, there is *no global clock*. Instead, the protocol Q depends only on the local clock of v (the number of time slots since its activation), and possibly additional information received earlier from other nodes.

There exists an important difference between the asynchronous wake-up studied in the wake-up problem, and the one defined in the unstructured radio network model. Specifically, in the wake-up problem, nodes are assumed to be woken up externally by messages. Depending on the specific application, either model may be more suitable. In the context of wireless ad hoc and sensor networks, however, the wake-up problem setting implies that all nodes are physically deployed at the same time. More importantly, the external wake-up necessitates the existence of low power "trigger" circuits, which operate continuously on small power budgets, and wake up more power-hungry circuits only upon receipt of a suitable signal from a neighboring node. Unfortunately, currently available standard hardware such as the Mica2 [123] wireless sensor nodes do not offer this functionality.

In a single-hop radio network, the wake-up problem is solved—that is, all nodes are woken up—as soon as one node was able to send successfully. For the case when n is globally known, [104] presented a randomized algorithm that, for any given $\epsilon > 0$, completes the wake-up process in time $O(n \log(1/\epsilon))$, with probability at least $1 - \epsilon$. This result has subsequently been improved to $O(\log n \log(1/\epsilon))$ by Jurdziński and Stachowiak [138]. In the same paper, the authors additionally present a corresponding lower bound of $\Omega(\log n \log(1/\epsilon)/(\log\log n + \log\log(1/e)))$, a bound that has recently been improved to $\Omega(\log n \log(1/e))$ in [86]. This last result is particularly interesting because it proves our MIS algorithm of Chapter 13 to be asymptotically optimal. In the case of n being unknown to the nodes, [138] has proven an almost linear time lower bound of $\Omega(n/\log n)$, thus establishing an exponential gap between these two models. The generalized wake-up problem in multi-hop radio network was first studied in [52]. Finally, the study of *deterministic* protocols for the wake-up problem has lead to efficient constructions of combinatorial structures called *radio synchronizers* [52, 104, 128] and *universal radio synchronizers* [49, 51].

While communication primitives such as broadcast, wake-up, or gossiping, have thus been extensively studied in the literature on radio networks in both randomized and deterministic versions, much less is known about the computation of local network coordination structures such as *clusterings* or *colorings* in the radio network model. This is somewhat surprising considering the particular importance of such structures in multi-hop wireless ad hoc and sensor networks. In fact, solving such problems in radio network models was mainly investigated in the context of backbone formation in wireless ad hoc and sensor networks, e.g. [99, 227]. However, these algorithms either have linear running time [227], or are based on strong assumptions, e.g. that nodes wake up at the same time, or that every node knows its one-hop or even two-hop neighborhood when waking up [99].

In a complementing thread of research, the (centralized) off-line *approximability* of computing short broadcast schedules has been studied. In this setting, the network topology is the input to a sequential algorithm and the goal is to come up with as short a feasible schedule as possible. This problem was shown to be NP-complete in [45] and—for Euclidean instances—in [211]. Approximation algorithms with a multiplicative approximation ratio of $O(\log^2 n)$ have first been presented in [30, 46]. In [80], Elkin and

Kortsarz presented an efficient construction of a broadcast schedule with additive approximation ratio $O(\log^4 n)$, a result that has recently been improved to an additive $O(\log^2 n)$-approximation by Gąsieniec, Peleg, and Xin in [105]. As for lower bounds, Elkin and Kortsarz prove multiplicative $\Omega(\log n)$ [78] and additive $\Omega(\log^2 n)$ [79] inapproximability, assuming that $NP \not\subseteq BPTIME(n^{O(\log \log n)})$. Finally, it was shown recently that under reasonable complexity assumptions, it is impossible to approximate the maximal number of new nodes that can be reached in a single round of a radio broadcast within a factor of $\Omega(\log^c n)$, for some constant $0 < c \leq 1$ [68].

Chapter 12

Coloring Unstructured Radio Networks

In this chapter, we study the construction of an initial *vertex coloring* that is useful for subsequent network organization tasks. A proper vertex coloring of a graph $G = (V, E)$ is an assignment of a color, denoted by color_v, to each node $v \in V$, such that any two adjacent nodes have a different color. The importance of such colorings in wireless ad hoc and sensor networks stems from the fact that when associating different colors with different time slots in a time-division multiple access (TDMA) scheme; a correct coloring corresponds to a medium access control (MAC) layer without *direct interference*, that is, no two neighboring nodes send at the same time. It is well known that in order to guarantee an entirely collision-free schedule in wireless networks, a correct vertex coloring is not sufficient. What is needed is a coloring of the *square* of the graph, i.e., a valid distance 2-coloring [205]. However, besides being a non-trivial first step towards obtaining a distance 2-coloring, a simple vertex-coloring ensures a schedule in which a receiver can be disturbed by at most (a small) constant number of interfering senders in a given time slot.

In the message passing model, there exists a long and rich body of work that deals with the distributed complexity of coloring problems, e.g., [91, 108, 109, 110, 166, 188]. In single-hop radio networks, the coloring problem has been studied in the form of the so-called *initialization problem* in which nodes must be assigned identifiers 1 through n [182, 183]. Typically, results on this problem are obtained in highly synchronized models in which each node has access to a global clock. In comparison, much less is known about coloring in *multi-hop radio networks*. In fact, the only previous algorithm for coloring in multi-hop radio networks appears to be the algorithm by Gandhi and Parthasarathy in [99]. This algorithm computes a correct distance-2 coloring using $O(\Delta)$ colors in time $O(\Delta \log^2 n)$ in a model that assumes synchronous wake-up.

In this chapter, we present and analyze an algorithm that computes a

vertex coloring in the unstructured radio network. Specifically, the algorithm uses $O(\Delta)$ colors and with high probability, every node can decide on a color within time $O(\Delta \log n)$ after its wake-up. Note that in a graph with bounded independence (e.g., unit disk graph), any correct coloring requires $\Omega(\Delta)$ colors. Also, the number of time slots required until every node can send at least one message to some other node in the network (and thus make its existence known to at least one node in the network) is at least $\Omega(\Delta)$, even if sender and receiver pairs are selected optimally.

As we will see, asynchronous wake-up of nodes has a big impact on the design of algorithms because it renders the symmetry breaking task significantly harder. It is easy to show that if every node transmits with a probability of $1/\Delta$, the time until every node can transmit collision-free to all its neighbors at least once is bounded by $O(\Delta \log n)$ (cf Lemma 12.4 of Section 12.2). With this observation, it is clear that any coloring algorithm in the message passing model can be transformed to a radio network model with synchronous wake-up. Specifically, the above simulation transforms any time T coloring algorithm in the message passing model into an $O(T \cdot \Delta \log n)$ time algorithm in a radio network model. In combination with the results presented in Chapter 8, this immediately yields coloring algorithms for radio networks with a running time of $O(\Delta \log \Delta \cdot \log n \log^* n)$ (or $O(\Delta \log n \log^* n)$, if distances can be measured). Apart from improving the running time compared to these simulations, the coloring algorithm presented in this chapter is capable of dealing with asynchronous wake-up.

12.1 Algorithm

During the algorithm, each node can be in various *states*. At any point in time, a node is in exactly one state, i.e., the sets of nodes induced by the different states form a partition of V.

- $\mathcal{Z}$: Nodes before their waking up. Sleeping nodes do not take part in the algorithm.
- $\mathcal{A}_i$: Nodes that are verifying (i.e., trying to decide on) color i.
- $\mathcal{R}$: Nodes that are requesting an *intra-cluster color* from their leader.
- $\mathcal{C}_i$: Nodes that have already irrevocably decided on color i.

The state $\mathcal{C}_0$ plays a special role and nodes in state $\mathcal{C}_0$ are called *leaders*. The algorithm itself is divided into three subroutines: Algorithm 12.1 for nodes in states $\mathcal{A}_i$, Algorithm 12.2 for nodes in state $\mathcal{R}$, and Algorithm 12.3 for nodes in state $\mathcal{C}_i$. The sequence of states that a node can be part of during the course of the algorithm is shown in Figure 12.1. A solid arrow represents the state transition a node makes when the event denoted by the arrow's label occurs. A dashed arrow between two states indicates the message type which is significant for the communication between nodes in these two states. Note that in our model, however, *every* neighbor of a sending node—regardless of their current state—may actually receive the message or experience a collision. Upon waking up, each node starts in state $\mathcal{A}_0$, without having any

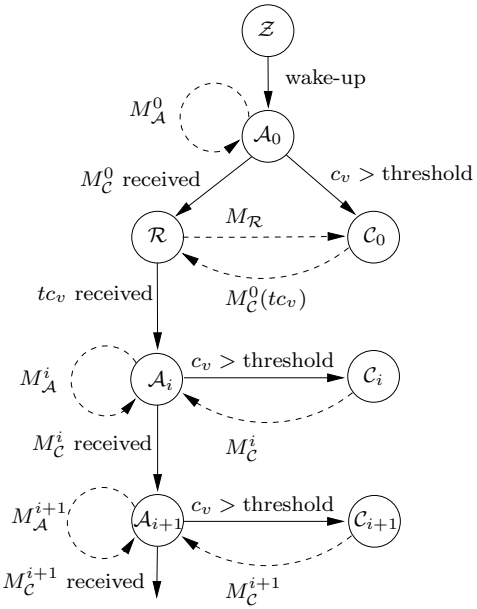

Figure 12.1: Sequence of states in the algorithm. Each color i is represented by a state $\mathcal{C}_i$, which a node enters at the moment it (irrevocably) selects color i. Before deciding on i, a node first has to *verify* (or compete for) i in state $\mathcal{A}_i$. If the node does not prevail in this verification, it moves from $\mathcal{A}_i$ to a new state $\mathcal{A}_{suc}$, which corresponds to either the intermediate requesting state $\mathcal{R}$ or the verification state of the next higher color $\mathcal{A}_{i+1}$ (cf Lines 3 and 15 of Algorithm 12.1).

knowledge whether some of its neighbors have already started executing the algorithm beforehand.

From a global point of view, the algorithm's main idea can be described easily: In a first stage, the nodes elect a set of mutually independent *leaders* (nodes in state $\mathcal{C}_0$) among themselves and each non-leader associates itself with a leader within its neighborhood. Since leaders are independent, they can safely assign themselves color 0. The set of leaders naturally induces *clusters* consisting of all nodes associated with the same leader. The task of each leader is to assign a unique *intra-cluster color* tc_v to every node v within its cluster. Unfortunately, the coloring induced by these intra-cluster colors may not form a valid coloring since two neighboring nodes in different clusters may be assigned the same intra-cluster color.

On the other hand, if the set of leaders is really independent, there can only be a small number of neighboring nodes with the same intra-cluster

color. The coloring induced by these intra-cluster colors thus represents a first coarse structuring of the network that facilitates the subsequent task of actually assigning colors to nodes. Technically, upon receiving an intra-cluster color tc_v from its leader, a node goes on to verify a specific color $tc_v(\kappa_2 + 1)$ against neighboring nodes from different clusters that may have received the same intra-cluster color. This *verification procedure* must ensure that no two neighboring nodes end up selecting this specific color.

The algorithmic difficulty of the above process stems from the fact that nodes wake up asynchronously and do not have access to a global clock. Therefore, the different phases (verification, requesting intra-cluster color, etc...) of different nodes may be arbitrarily intertwined or shifted in time. While some nodes may still compete for becoming leader in state $\mathcal{A}_0$, their neighbors may already be much more advanced in their coloring process. Moreover, messages may be lost due to collisions at any time. In view of this harsh environment, the primary challenge is that the algorithm must achieve two contradictory aims: symmetries between nodes must be broken both *correctly and rapidly*. That is, no two neighboring nodes ever select the same color and yet, every node can take its decision shortly after its wake-up (i.e., there is no starvation).

A crucial part of reconciling these contradictory aims takes place in the verification procedure. In order to ensure both correctness and fast progress in all parts of the network with high probability, our algorithm uses a technique of counters, critical ranges, and local competitor lists. Roughly, the idea is that every node v uses a local counter c_v which it increments in every time slot. Intuitively, this counter represents v's progress towards deciding on color i and v selects i as soon as c_v reaches a certain threshold.

In order to prevent two neighboring nodes from selecting the same color, the algorithm must make sure that as soon as a node v selects its color, all neighbors of v can be notified before their counter also reaches the threshold. In view of collisions and message losses being always possible, there must be a large enough time interval between two neighboring counters reaching the threshold. A simple idea to achieve this correctness condition is to have every node transmit its current counter with a certain sending probability. Whenever a node receives a message with higher counter, it resets its own counter. Unfortunately, this technique may lead to *chains of cascading resets*, i.e., a node's counter is reset by a more advanced node, which in turn is reset by another node and so forth. While, eventually, one node will end up selecting the color, this method does not prevent nodes from starving in certain (local) parts of the network graph.

Our algorithm therefore employs a more subtle handling of the counters. The general idea is that upon receiving a message from a neighbor, a node only resets its counter if it is within a *critical range* of the received counter (see Line 18 of Algorithm 12.1). On the one hand, this critical range is large enough to ensure correctness with high probability. On the other hand, this technique allows for much more parallelism in the network because many nodes can simultaneously make progress towards deciding on the color. In order to truly avoid cascading resets and achieve the claimed running time, however, using only counters and critical ranges is insufficient. Specifically,

nodes should also be prevented from resetting their counter to a value within the critical range of neighboring nodes and furthermore, all counters must remain relatively close to the verification threshold even after a reset. For this purpose, each node stores a local *competitor list* containing the current counter values of neighboring nodes.

Unfortunately, in the unstructured radio network model, it is impossible to constantly keep this competitor list and the corresponding locally stored counters complete and correctly updated. Interestingly, we can prove in Section 12.2 that in spite of this inevitable inconsistency, our technique of using counters and critical ranges in combination with storing local competitor lists avoids cascading resets and at the same time ensures the correctness condition. That is, whenever a node v selects a color, the counters of all neighboring competing nodes are far enough from the threshold so that v has enough time to inform its neighbors with high probability.

upon entering state $\mathcal{A}_i$: (when waking up, a node is initially in state $\mathcal{A}_0$)

1: $P_v := \emptyset$; {* v is passive *}
2: $\zeta_i := \begin{cases} 1 & , i = 0 \\ \Delta & , i > 0 \end{cases}$
3: $\mathcal{A}_{suc} := \begin{cases} \mathcal{R} & , i = 0 \\ \mathcal{A}_{i+1} & , i > 0 \end{cases}$
4: **for** $\lceil \alpha\Delta \log n \rceil$ time slots **do**
5: **for each** $w \in P_v$ **do** $d_v(w) := d_v(w) + 1$;
6: **if** $M^i_{\mathcal{A}}(w, c_w)$ received **then** $P_v := P_v \cup \{w\}$; $d_v(w) := c_w$; **end if**
7: **if** $M^i_{\mathcal{C}}(w)$ received **then** state := $\mathcal{A}_{suc}$; $L(v) := w$; **end if**
8: **end for**
9: $c_v := \chi(P_v)$, where $\chi(P_v)$ is the maximum value such that, $\chi(P_v) \leq 0$ and $\chi(P_v) \notin [c_w - \lceil \gamma\zeta_i \log n \rceil, \ldots, c_w + \lceil \gamma\zeta_i \log n \rceil]$ for each $w \in P_v$;
10: **while** *state* $= \mathcal{A}_i$ **do** {* v is active *}
11: $c_v = c_v + 1$;
12: **for each** $w \in P_v$ **do** $d_v(w) := d_v(w) + 1$;
13: **if** $c_v \geq \lceil \sigma\Delta \log n \rceil$ **then** state := $\mathcal{C}_i$; **end if**
14: **transmit** $M^i_{\mathcal{A}}(v, c_v)$ with probability $1/(\kappa_2\Delta)$;
15: **if** $M^i_{\mathcal{C}}(w)$ received **then** state := $\mathcal{A}_{suc}$; $L(v) := w$; **end if**
16: **if** $M^i_{\mathcal{A}}(w, c_w)$ received **then**
17: $P_v := P_v \cup \{w\}$; $d_v(w) := c_w$;
18: **if** $|c_v - c_w| \leq \lceil \gamma\zeta_i \log n \rceil$ **then** $c_v := \chi(P_v)$; **end if**
19: **end if**
20: **end while**

Algorithm 12.1: Coloring Algorithm—Node v in state $\mathcal{A}_i$

In more detail, the algorithm works as follows. Upon waking up, a node enters state $\mathcal{A}_0$ and tries to become a leader. Generally, whenever a node v enters a state $\mathcal{A}_i$, for $i \geq 0$, it first waits for $\lceil \alpha\Delta \log n \rceil$ time slots. As

```
upon entering state R:
 1: while state = R do          {* v is active *}
 2:    transmit M_R(v, L(v)) with probability 1/(κ2 Δ);
 3:    if M_C^0(L(v), v, tc_v) received then
 4:       state := A_{tc_v · (κ2+1)};
 5:    end if
 6: end while
```

Algorithm 12.2: Coloring Algorithm—Node v in state $\mathcal{R}$

```
upon entering state C_i:
 1: color_v := i;          {* v is active *}
 2: if i > 0 then
 3:       repeat forever transmit M_C^i with probability 1/(κ2 Δ);
 4: else if i = 0 then
 5:       tc := 0;
 6:       Q := ∅;    {FIFO request queue }
 7:       repeat forever
 8:            if M_R(w, v) received and w ∉ Q then add w to Q; end if
 9:            if Q is empty then
10:                 transmit M_C^0(v) with probability 1/κ2;
11:            else
12:                 tc := tc + 1;
13:                 Let w be first element in Q;
14:                 for ⌈β log n⌉ time slots do
15:                      transmit M_C^0(v, w, tc) with probability 1/κ2;
16:                 end for
17:                 Remove w from Q;
18:            end if
19:       end repeat
20: end if
```

Algorithm 12.3: Coloring Algorithm—Node v in state $\mathcal{C}_i$

soon as it receives a messages $M_{\mathcal{C}}^i$ from a neighboring node that has already joined $\mathcal{C}_i$ (Line 7 of Algorithm 12.1), v joins the succeeding state $\mathcal{A}_{suc}$, which corresponds to $\mathcal{R}$ in the case $i = 0$, and $\mathcal{A}_{i+1}$, otherwise. If no such message is received, v becomes *active* and starts competing for color i (Line 10).

In order to ensure with high probability that no two neighbors enter the same state $\mathcal{C}_i$, the following process is employed: In each time slot, an active node $v \in \mathcal{A}_i$ increments its counter c_v and transmits a message $M_{\mathcal{A}}^i$ with probability $1/(\kappa_2\Delta)$ (Lines 11 and 14, respectively). Whenever v receives a message $M_{\mathcal{C}}^i$ from a neighbor $w \in \mathcal{C}_i$, v knows that it cannot verify color i anymore and consequently moves on to state $\mathcal{A}_{suc}$.

When receiving a message $M_{\mathcal{A}}^i(w, c_w)$ from a neighboring competing node $w \in \mathcal{A}_i$, v adds neighbor w to its *competitor list* P_v and stores a *local copy* of w's counter c_w denoted by $d_v(w)$ (Line 17). In each subsequent time slot, these local copies $d_v(w)$ are incremented in order to keep track with

the real current counter of w as much as possible. Moreover, in Line 18, v compares c_w to its own counter c_v. If the two counters are within the *critical range* $\lceil \gamma\zeta_i \log n \rceil$ of each other, v resets its own counter to $\chi(P_v)$. The value $\chi(P_v) < 0$ (Line 9) is defined such that the new counter is not within the critical range $\lceil \gamma\zeta_i \log n \rceil$ of any locally stored copy of neighboring counters. Notice, however, that because counters may be reset in any time slot, a locally stored copy $d_v(w)$ of c_w may be outdated without v knowing it. For instance, if w has to reset its counter due to receipt of a message $M^i_{\mathcal{A}}(x, c_x)$ from a neighbor x, and if v does not receive this message (possibly due to a collision or because x and v are not neighbors), it subsequently holds $d_v(w) \neq c_w$. Hence, in spite of the definition of $\chi(P_v)$, a node's counter may be within the critical range of a neighboring counter after a reset.

If in the above process, a node succeeds in incrementing its counter up to the threshold of $\lceil \sigma\Delta \log n \rceil$ (Line 13), it decides on color i and joins state $\mathcal{C}_i$. As mentioned before, the technique of using counters and critical ranges guarantees that quick progress is made simultaneously in all parts of the network. Specifically, this method ensures that after a limited (constant) number of trials, at least one competing node in $\mathcal{A}_i$ can join $\mathcal{C}_i$ in *every region* of the graph. At the same time, the method also guarantees with high probability that no two neighboring nodes join $\mathcal{C}_i$, i.e., the set of nodes induces by $\mathcal{C}_i$ is independent.

A special role in the algorithm plays the state $\mathcal{C}_0$, the set of leaders. A leader's duty is to assign unique intra-cluster colors to each node in its cluster. Specifically, each non-leader v in $\mathcal{R}$ assigned to leader w sends requests $M_{\mathcal{R}}(v, w)$ for an intra-cluster color to w. Upon receiving such a request message from v, w transmits for $\lceil \beta \log n \rceil$ time slots with probability $1/\kappa_2$ a message $M^0_{\mathcal{C}}(w, v, tc)$, where tc denotes the intra-cluster color assigned to v and is incremented for each subsequent requesting node. If necessary, requests are buffered in an internal queue $\mathcal{Q}$, which helps in keeping all messages within the size of $O(\log n)$ bits.

In state $\mathcal{R}$, a non-leader node v requests an intra-cluster color from its leader. As soon as v receives a message $M^0_{\mathcal{C}}(w, v, tc_v)$ from leader w containing its intra-cluster color tc_v, v moves on to state $\mathcal{A}_{tc_v(\kappa_2+1)}$, i.e., it attempts to verify color $c = tc_v(\kappa_2 + 1)$ next. If verifying color c is unsuccessful (i.e., if a neighbor of v selects color c earlier), a node joins the next higher state $\mathcal{A}_{suc} = \mathcal{A}_{tc_v(\kappa_2+1)+1}$, and so forth, until it manages to verify and decide on a color. In Corollary 12.7 of Section 12.2, we show that every node is capable of deciding on a color from $tc_v(\kappa_2 + 1), tc_v(\kappa_2 + 1) + 1, \ldots, tc_v(\kappa_2 + 1) + \kappa_2$ with high probability. Hence, the reason for a node to verify $\mathcal{A}_{tc_v(\kappa_2+1)}$ upon receiving tc_v is that by doing so, two nodes with different intra-cluster colors never compete for the same color. This turns out to be an important ingredient when upper bounding the amount of time each node must maximally wait before deciding on its color.

The algorithm's four parameters, α, β, γ, and σ can be chosen as to trade-off the running time and the probability of correctness. The higher the parameters, the less likely the algorithm fails in producing a correct coloring. In order to obtain the high probability result in Section 12.2, the parameters

are defined as $\alpha \geq 2\gamma\kappa_2 + \sigma + 1$, $\beta \geq \gamma$, and

$$\gamma = \frac{5\kappa_2}{\left[\frac{1}{e}\left(1-\frac{1}{\kappa_2}\right)\right]^{\frac{\kappa_1}{\kappa_2}}\left[\frac{1}{e}\left(1-\frac{1}{\kappa_2\Delta}\right)\right]^{\frac{1}{\kappa_2}}}, \quad \sigma = \frac{10e^2\kappa_2}{\left(1-\frac{1}{\kappa_2}\right)\left(1-\frac{1}{\kappa_2\Delta}\right)},$$

for $\Delta \geq 2$. Specifically, these constants guarantee a correct coloring and running time with probability at least $1 - O(n^{-1})$. Simulation results show that in randomly distributed networks significantly smaller constants suffice, yielding a practically efficient coloring algorithm.

12.2 Analysis

In this section, we prove that the algorithm of Section 12.1 is both correct and complete with high probability. *Correctness* means that no two adjacent nodes end up having the same color, *completeness* leaves no node without a color. Furthermore, we show that every node decides on a color after at most $O(\Delta \log n)$ time slots for constant κ_2. For clarity of exposition, we omit ceiling signs in the analysis, i.e, we consider all non-integer values to be implicitly rounded to the next higher integer value. Further, let $c_v(t)$ be the value of the counter of node v at time t. We call a node in $\mathcal{A}_i$ *covered* if either itself or one of its neighbors is in $\mathcal{C}_i$.

For future reference, we begin with a simple lemma that bounds the maximum number of nodes in the 2-hop neighborhood of any node.

Lemma 12.1. *Let $G = (V, E)$ be a graph with at most κ_2 independent nodes in the 2-hop neighborhood of any node. It follows that every node has at most $\kappa_2\Delta$ 2-hop neighbors.*

Proof. Every node has at most κ_2 mutually independent nodes in its 2-hop neighborhood, and each such node has at most Δ neighbors. □

We now state two lemmas that give us probabilistic bounds on the amount of time required until a message is correctly transmitted from a sender v to an intended receiver u in the algorithm. Notice that both lemmas hold only under the assumption that the set $\mathcal{C}_0$ of leaders forms a correct independent set.

Lemma 12.2. *Assume $\mathcal{C}_0$ forms an independent set. Consider two neighboring nodes u and v and let I be a time interval of length $\gamma\Delta \log n$. If v is active throughout the interval I, u receives at least one message from v during I with probability $1 - n^{-5}$.*

Proof. Let p_v denote the transmission probability of v. Recall that nodes in C_0 transmit with a probability of $1/\kappa_2$, whereas the sending probability of

all other nodes is $1/(\kappa_2\Delta)$. The probability P_s that v succeeds in sending a message to u in a time slot $t \in I$ is

$$\begin{aligned} P_s &= p_v \prod_{i\in\Gamma(u)\setminus\{v\}} (1-p_i) \geq p_v \prod_{i\in\Gamma(u)\cap\mathcal{C}_0} (1-p_i) \prod_{j\in\Gamma(u)\setminus\mathcal{C}_0} (1-p_j) \\ &\geq p_v \left(1-\frac{1}{\kappa_2}\right)^{\kappa_1} \left(1-\frac{1}{\kappa_2\Delta}\right)^{\Delta} \\ &> p_v \left[\frac{1}{e}\left(1-\frac{1}{\kappa_2}\right)\right]^{\frac{\kappa_1}{\kappa_2}} \left[\frac{1}{e}\left(1-\frac{1}{\kappa_2\Delta}\right)\right]^{\frac{1}{\kappa_2}}, \end{aligned} \tag{12.1}$$

where the last inequality follows from Fact 2.2 and $\kappa_2 \geq \kappa_1$. Because v is assumed to be active throughout the interval I and for every active node $p_v \geq 1/(\kappa_2\Delta)$, the probability P_{no} that u does not receive a message from v during I is

$$\begin{aligned} P_{no} &= (1-P_s)^{|I|} \\ &< \left(1-\frac{1}{\kappa_2\Delta}\left[\frac{1}{e}\left(1-\frac{1}{\kappa_2}\right)\right]^{\frac{\kappa_1}{\kappa_2}} \left[\frac{1}{e}\left(1-\frac{1}{\kappa_2\Delta}\right)\right]^{\frac{1}{\kappa_2}}\right)^{\gamma\Delta\log n} \\ &\underset{\text{Fact 2.2}}{\leq} n^{-\frac{\gamma}{\kappa_2}\left[\frac{1}{e}\left(1-\frac{1}{\kappa_2}\right)\right]^{\kappa_1/\kappa_2}\left[\frac{1}{e}\left(1-\frac{1}{\kappa_2\Delta}\right)\right]^{1/\kappa_2}} < n^{-5}, \end{aligned}$$

where the last inequality follows from the definition of γ. □

Lemma 12.3. *Assume $\mathcal{C}_0$ forms an independent set. Consider two neighboring nodes u and $v \in C_0$ and let I' be a time interval of length $\gamma \log n$. If $v \in C_0$ throughout the interval I', u receives at least one message from v during I' with probability $1 - n^{-5}$.*

Proof. The proof is virtually identical to the previous one. In the case $v \in \mathcal{C}_0$, it holds that $p_v = 1/\kappa_2$ and plugging this value into Inequality (12.1) and applying Fact 2.2 yields

$$\begin{aligned} P_{no} &= (1-P_s)^{|I'|} < \left(1-\frac{1}{\kappa_2}\left[\frac{1}{e}\left(1-\frac{1}{\kappa_2}\right)\right]^{\frac{\kappa_1}{\kappa_2}} \left[\frac{1}{e}\left(1-\frac{1}{\kappa_2\Delta}\right)\right]^{\frac{1}{\kappa_2}}\right)^{\gamma\log n} \\ &\underset{\text{Fact 2.2}}{<} n^{-5}. \end{aligned}$$

□

For the next lemma, we first define the notion of a *successful transmission*. A node v transmits *successfully* in a time slot t if *all* nodes $u \in \Gamma(v) \setminus \{v\}$ within the transmission range of v (i.e., in v's 1-hop neighborhood) receive the message without collision. In the following lemma, we show that with high probability, at least one node in v's neighborhood can transmit *successfully* during any interval of length $O(\kappa_2\Delta\log n)$.

Lemma 12.4. *Assume $\mathcal{C}_0$ forms an independent set. Consider a node $v \in \mathcal{A}_i$ for an arbitrary i. Further, let I be a time interval of length $|I| = \frac{\sigma}{2}\Delta \log n$ during which $v \in \mathcal{A}_i$ is active. With probability $1 - n^{-5}$, there is at least one time slot $t \in I$ such that a node $u \in \Gamma(v) \cap \mathcal{A}_i$ transmits* successfully.

Proof. By Lemma 12.1, there are at most $\kappa_2\Delta$ nodes in the 2-neighborhood of any node. If in a time slot, a node w is the only transmitting node in $\Gamma_2(w)$, it is guaranteed that w transmits *successfully* because no node outside $\Gamma_2(w)$ can cause a collision at a neighbor of w. Define P_s to be the probability that a node $w \in \Gamma(v) \cap \mathcal{A}_i$ transmits successfully in a given time slot $t \in I$. By the above argument, P_s is lower bounded by

$$\begin{aligned}
P_s &\geq \sum_{w \in \Gamma(v) \cap \mathcal{A}_i} \left(p_w \cdot \prod_{\substack{u \in \Gamma_2(w) \\ u \neq w}} (1 - p_u) \right) \\
&\geq \sum_{w \in \Gamma(v) \cap \mathcal{A}_i} p_w \cdot \prod_{u \in \Gamma_2(w) \setminus \mathcal{C}_0} \left(1 - \frac{1}{\kappa_2\Delta}\right) \cdot \prod_{u \in \Gamma_2(w) \cap \mathcal{C}_0} \left(1 - \frac{1}{\kappa_2}\right) \\
&\geq \sum_{w \in \Gamma(v) \cap \mathcal{A}_i} p_w \cdot \left(1 - \frac{1}{\kappa_2\Delta}\right)^{\kappa_2\Delta} \left(1 - \frac{1}{\kappa_2}\right)^{\kappa_2} \\
&\underset{\text{Fact 2.2}}{\geq} \frac{1}{e^2\kappa_2\Delta} \left(1 - \frac{1}{\kappa_2\Delta}\right) \left(1 - \frac{1}{\kappa_2}\right)
\end{aligned}$$

because $\sum_{w \in \Gamma(v) \cap \mathcal{A}_i} p_w$ is at least $1/(\kappa_2\Delta)$ for as long as v is active in $\mathcal{A}_i$. Finally, the probability P_{no} that no node in $\Gamma(v) \cap \mathcal{A}_i$ manages to transmit successfully within the interval I during which v is active in $\mathcal{A}_i$ is

$$\begin{aligned}
P_{no} &= (1 - P_s)^{|I|} \leq \left(1 - \frac{1}{e^2\kappa_2\Delta} \left(1 - \frac{1}{\kappa_2\Delta}\right) \left(1 - \frac{1}{\kappa_2}\right)\right)^{\frac{\sigma}{2}\Delta \log n} \\
&\underset{\text{Fact 2.2}}{\leq} e^{-\frac{\sigma}{2e^2\kappa_2} \frac{\kappa_2 - 1}{\kappa_2} \frac{\kappa_2\Delta - 1}{\kappa_2\Delta} \log n} < n^{-5}.
\end{aligned}$$

The last step follows from the definition of σ. □

Lemmas 12.2, 12.3, and 12.4 are based on the assumption that the set of leaders $\mathcal{C}_0$ forms an independent set. Therefore, in order to make full use of these lemmas, we need to prove that this assumption holds for the entire duration of the algorithm. Intuitively, the reason for our claim is the following. By the definition of the algorithm, only nodes in state $\mathcal{A}_0$ can enter state $\mathcal{C}_0$. If such a candidate node $v \in \mathcal{A}_0$ transmits *successfully*, all neighboring nodes $w \in \Gamma(v) \cap \mathcal{A}_0$ having a counter value within the *critical range* $\gamma\zeta_0 \log n = \gamma \log n$ of v's counter will reset their counter to $\chi(P_w)$, which is by definition outside the critical range of v. Hence, once node v was able to transmit successfully, no neighboring candidate node

$w \in \Gamma(v) \cap \mathcal{A}_0$ can block v from incessantly incrementing its counter until it reaches the threshold $\sigma\Delta \log n$ which enables to join $\mathcal{C}_0$. The only way v can still be prevented from becoming a leader is if v receives a message $M_{\mathcal{C}}^0$ from a neighbor that has entered $\mathcal{C}_0$ before v's counter reaches the threshold. Moreover, the counters of neighboring nodes being outside the critical range, it can be shown that upon becoming leader, v has enough time to inform all neighbors of its having joined $\mathcal{C}_0$.

We formalize this intuition in Lemma 12.5 and its subsequent proof. More precisely, the lemma proves the more general statement that every color class $\mathcal{C}_i$ (i.e., not merely $\mathcal{C}_0$) forms an independent set at all times during the algorithm's execution with high probability. Notice that the lemma establishes the algorithm's *correctness*, because if all color classes form independent sets, the resulting coloring is necessarily correct.

Lemma 12.5. *For all i, the color class $\mathcal{C}_i$ forms an independent set throughout the execution of the algorithm with probability at least $1 - 2n^{-3}$.*

Proof. At the beginning, when the first node wakes up, the claim certainly holds, because $\mathcal{C}_i = \emptyset$ for all i. We will now show that with high probability the claim continues to hold throughout the algorithm's execution. For this purpose, consider an arbitrary node $v \in \mathcal{A}_i$ and assume for contradiction that v is the *first node* to violate the independence of $\mathcal{C}_i$ for an arbitrary $i \geq 0$. That is, we assume that v is the first node to enter $\mathcal{C}_i$ even though a neighboring node w has entered $\mathcal{C}_i$ in the same or a previous time slot. Note that if two or more nodes violate the independence of $\mathcal{C}_i$ simultaneously, we consider each of them to be the *first node*. We prove in the sequel that the probability of v being such a *first node* for a specific $\mathcal{C}_i$ is at most $2n^{-5}$. Applying the union bound, we conclude that the probability that there exists a node $v \in V$ that violates the independence of some $\mathcal{C}_i$ is bounded by $n^2 \cdot 2n^{-5} = 2n^{-3}$.

Let t_v^* be the time slot in which v enters state $\mathcal{C}_i$, $i \geq 0$. Since v is among the *first* nodes to violate the independence of any $\mathcal{C}_i$, and hence also $\mathcal{C}_0$, we know that for all time slots $t < t_v^*$, $\mathcal{C}_0$ is a correct independent set. That is, if v is among the first nodes to create a violation, Lemmas 12.2, 12.3, and 12.4 can be applied until time slot $t_v^* - 1$.

Let w be a neighbor of v that has joined $\mathcal{C}_i$ before v (or in the same time slot as v), say at time $t_w^* \leq t_v^*$. We consider two cases, $t_w^* < t_v^* - \gamma\zeta_i \log n$ and $t_w^* \geq t_v^* - \gamma\zeta_i \log n$, and start with the former.

If $t_w^* < t_v^* - \gamma\zeta_i \log n$, then w entered state $\mathcal{C}_i$ at least $\gamma\zeta_i \log n$ time slots before v. By Lemma 12.2 ($i > 0$) or Lemma 12.3 ($i = 0$), the probability that w manages to successfully send a message $M_{\mathcal{C}}^i$ to v during these $\gamma\zeta_i \log n$ time slots (during which v must be in $\mathcal{A}_i$ if it joins $\mathcal{C}_i$ at time t_v^*) is at least $1 - n^{-5}$. By Line 15 of Algorithm 12.1, however, v leaves state $\mathcal{A}_i$ and moves on to state $\mathcal{A}_{suc}$ upon receiving $M_{\mathcal{C}}^i$, i.e., it does not enter $\mathcal{C}_i$.

For the second case, we compute the probability that v joins $\mathcal{C}_i$ within $\gamma\zeta_i \log n$ time slots after t_w^*. Recall that by the definition of the algorithm, it holds that $c_w = \sigma\Delta \log n$ at time t_w^*. Consider the time interval I_w of length $\gamma\Delta \log n$ before t_w^*. Because in each time slot, counters of nodes in $\mathcal{A}_i$

are either incremented by one or set to $\chi(P_v) \leq 0$ and because $\sigma\Delta \log n > 2\gamma\Delta \log n$, it follows that c_w was not reset during I_w. If it was, c_w would not have reached $\sigma\Delta \log n$ by time t_w^*. Similarly, if c_v was reset during I_w, t_v^* could not be within $\gamma\zeta_i \log n$ of t_w^*. Hence, neither c_w nor c_v were reset during the interval I_w and it holds that at time t_w^*, $c_v \geq \sigma\Delta \log n - \gamma\zeta_i \log n$. More generally, it holds that

$$|c_w(t_w^* - h) - c_v(t_w^* - h)| \;\leq\; \gamma\zeta_i \log n$$

for each $h = 0, \ldots, \gamma\Delta \log n - 1$. By Lemma 12.2, the probability that v receives at least one message $M_{\mathcal{A}}^i$ from w during these $\gamma\Delta \log n$ time slots in I_w is at least $1 - n^{-5}$. If it does receive such a $M_{\mathcal{A}}^i$, v resets its counter (Line 18) and does not enter $\mathcal{C}_i$ within $\gamma\zeta_i \log n$ time slots of t_w^*.

Combining both cases, we know that with probability $1 - n^{-5}$, v does not enter $\mathcal{C}_i$ until $\gamma\zeta_i \log n$ time slots after its first neighbor has joined $\mathcal{C}_i$. And with probability $1 - n^{-5}$, v does not enter $\mathcal{C}_i$ thereafter. Consequently, the probability of v being a first node to violate the independence of a specific $\mathcal{C}_i$ is at most $2n^{-5}$. Each state $\mathcal{C}_i$ thus remains independent throughout the algorithm's execution with probability at least $1 - 2n^{-4}$. Finally, we can crudely upper bound the number of non-empty states $\mathcal{C}_i$ used in the algorithm by n, because in Lines 7 and 15, a node changes its state only if it has received a message $M_{\mathcal{C}}^i$ from a node that has already decided on $\mathcal{C}_i$. The probability that all color classes form independent sets at all times is at least $1 - 2n^{-3}$. □

Lemma 12.5 proves that with high probability, all color classes are independent and hence, the algorithm eventually produces a correct coloring. Particularly, notice that the lemma implies that the set of leaders $\mathcal{C}_0$ forms an independent set with high probability and hence, we can use Lemmas 12.2, 12.3, and 12.4 without restriction. What remains to be shown are the bounds on the running time as well as on the number of colors required. For this purpose, we first prove a helper lemma that bounds the number of nodes v that can simultaneously be in the same active state $\mathcal{A}_i$.

Lemma 12.6. *If the set of nodes in $\mathcal{C}_0$ is independent, then for any $i > 0$, the number of nodes in any 1-hop neighborhood that ever join state $\mathcal{A}_i$ is at most κ_2.*

Proof. A node v enters a state $\mathcal{A}_i$, $i > 0$, for the first time when being in state $\mathcal{R}$ and receiving a message $M_{\mathcal{C}}^0(w, v, tc_v)$ from its leader $w = L(v)$. Consider a leader $w \in \mathcal{C}_0$ and let S_w denote the set of nodes having w as their leader, i.e., $S_w = \{v \mid L(v) = w\}$. Since the value tc is incremented for every new node v in the queue $\mathcal{Q}$, w assigns to each $v \in S_w$ a unique *intra-cluster color* tc_v. While being unique within each cluster, these intra-cluster colors do not constitute a legal coloring, because neighboring nodes belonging to different clusters may be assigned the same intra-cluster color tc_v by their respective leaders. If the set of leaders $w \in \mathcal{C}_0$ forms an independent set, the maximum number of leaders $w \in \mathcal{C}_0$ in any 2-hop neighborhood is κ_2. Therefore, every

node can have at most κ_2 1-hop neighbors (including itself!) with the same intra-cluster color tc_v.

In Line 4 of Algorithm 12.2, a node v with tc_v enters state $\mathcal{A}_{tc_v(\kappa_2+1)}$. That is, two nodes with subsequent intra-cluster colors tc_v and tc_v+1 enter states $\mathcal{A}_i$ and $\mathcal{A}_j$, where $|i-j| = \kappa_2+1$. By the definition of Algorithm 12.1, the only way a node can move from state $\mathcal{A}_i$ to state $\mathcal{A}_{i+1}$ is by receiving a message $M^i_{\mathcal{C}}$ from a neighboring node that has already entered $\mathcal{C}_i$. Without receiving $M^i_{\mathcal{C}}$ a node will eventually join state $\mathcal{C}_i$ itself. Hence, whenever a node v in $\mathcal{A}_i$ moves on to state $\mathcal{A}_{i+1}$, at least one of its neighbors must have joined $\mathcal{C}_i$. From this, it follows that each of the at most κ_2 neighbors of a node v that are assigned the same intra-cluster color tc_v will decide on a color in the range $tc_v(\kappa_2+1), \ldots, tc_v(\kappa_2+1)+\kappa_2$. Notice that this range does not overlap with the corresponding range of the next higher intra-cluster color which starts with color $(tc_v+1)(\kappa_2+1) > tc_v(\kappa_2+1)+\kappa_2$. Consequently, nodes assigned to different intra-cluster colors are never in the same state $\mathcal{A}_i$ for any $i > 0$. And because at most κ_2 nodes are assigned the same tc_v in the 1-hop neighborhood of any node v, the lemma follows. □

The proof of Lemma 12.6 implicitly gives raise to the following corollary.

Corollary 12.7. *While executing the algorithm, every node v is at most in κ_2+1 different states $\mathcal{A}_i$, namely $\mathcal{A}_0, \mathcal{A}_{tc_v(\kappa_2+1)}, \ldots, \mathcal{A}_{tc_v(\kappa_2+1)+\kappa_2}$. This holds under the condition that the nodes in $\mathcal{C}_0$ are independent.*

The next lemma gives a lower bound on the counters c_v of any node $v \in \mathcal{A}_i$.

Lemma 12.8. *Let c_v be the counter of node $v \in \mathcal{A}_i$. It holds throughout the execution of the algorithm that $c_v \geq -2\gamma\Delta\log n - 1$, if $i = 0$, and $c_v \geq -2\kappa_2\gamma\Delta\log n - 1$, otherwise. This holds under the condition that the nodes in $\mathcal{C}_0$ are independent.*

Proof. Consider a node $v \in \mathcal{A}_i$. The only time v's counter c_v is set to a negative value is when (re)setting c_v to $\chi(P_v)$ in Lines 9 or 18 of Algorithm 12.1. $\chi(P_v)$ is defined as the largest value such that $\chi(P_v) < 0$ and $\chi(P_v) \notin [c_u - \gamma\zeta_i\log n, \ldots, c_u + \gamma\zeta_i\log n]$ for each $u \in P_v$. Because the set P_v contains only nodes that are also in state $\mathcal{A}_i$, it follows from Lemma 12.6 that $|P_v| \leq \kappa_2$ for any $i > 0$, if the nodes in $\mathcal{C}_0$ form an independent set. In the case $i = 0$, it trivially holds that $|P_v| \leq \Delta$.

The number of values that are prohibited for $\chi(P_v)$ is therefore at most $\kappa_2 \cdot 2\gamma\zeta_i\log n$ in the case $i > 0$ and $\Delta \cdot 2\gamma\zeta_0\log n$ if $i = 0$. Plugging in the values for ζ_i, we can write

$$\chi(P_v) \;\geq\; \begin{cases} -2\gamma\Delta\log n - 1 & , i = 0 \\ -2\kappa_2\gamma\Delta\log n - 1 & , i > 0 \end{cases},$$

which concludes the proof. □

Having the last two helper lemmas, we are now ready to analyze the algorithm's running time, that is, to bound the maximum amount of time between a node's waking up and its entering a color class $\mathcal{C}_i$. We first obtain a bound on the amount of progress achieved by nodes in a state $\mathcal{A}_i$ in every part of the graph.

Lemma 12.9. *Let T_v^i denote the number of time slots a node v spends in state $\mathcal{A}_i$. With probability $1 - 3n^{-3}$, it holds for all v and i that $T_v^i \in O(\kappa^3 \Delta \log n)$.*

Proof. By Lemma 12.5 implies that with probability $1 - 2n^{-3}$, the set of nodes in state $\mathcal{C}_0$ forms an independent set. In the sequel of the proof, we focus on this case and assume that all nodes in $\mathcal{C}_0$ are mutually independent.

Let t_v denote the time slot in which node $v \in \mathcal{A}_i$ executes Line 9 of Algorithm 12.1. Until t_v, v spends exactly $\alpha\Delta \log n$ time slots in $\mathcal{A}_i$. By Lemma 12.4, we know that at least one node $w \in \Gamma(v) \cap \mathcal{A}_i$ is able to transmit successfully during the interval $I = [t_v, t_v + \frac{\sigma}{2}\Delta \log n]$ with probability $1 - n^{-5}$ (unless v leaves state $\mathcal{A}_i$ during that interval in which case Lemma 12.9 clearly holds). Say this happens at time t_w^s. According to Lines 6 and 17 of Algorithm 12.1, all nodes $u \in \Gamma(w) \cap \mathcal{A}_i$ store a local copy $d_u(w)$ of w's current counter c_w upon receiving w's message $M_{\mathcal{A}}^i$ in time slot t_w^s. In Lines 5 and 12, this local copy is incremented by one in each subsequent time slot. That is, as long as w's real counter is not reset to $\chi(P_w)$, every node $u \in \Gamma(w) \cap \mathcal{A}_i$ has a correct local copy $d_u(w)$ of w's current counter c_w.

We now show that w's counter c_w cannot be reset by any node $u \in \Gamma(w) \cap \mathcal{A}_i$ after t_w^s anymore. First, in Line 18, every node $u \in \Gamma(w) \cap \mathcal{A}_i$ whose counter $c_u(t_w^s)$ at time t_w^s is in the range

$$[c_w(t_w^s) - \gamma\zeta_i \log n, \ldots, c_w(t_w^s) + \gamma\zeta_i \log n]$$

resets its own counter to $\chi(P_u)$ in time slot t_w^s. Recall that $\chi(P_u)$ is defined as the maximum value such that $\chi(P_u) \leq 0$ and $\chi(P_u) \notin [c_x - \gamma\zeta_i \log n, \ldots, c_x + \gamma\zeta_i \log n]$ for each $x \in P_u$. Specifically, because w transmitted successfully, this means that $\chi(P_u) \notin [c_w - \gamma\zeta_i \log n, \ldots, c_w + \gamma\zeta_i \log n]$, and hence $|c_u(t_w^s + 1) - c_w(t_w^s + 1)| > \gamma\zeta_i \log n$. Clearly, the same inequality also holds for all nodes $u \in \Gamma(w) \cap \mathcal{A}_i$ whose counter was not in the critical range $[c_w(t_w^s) - \gamma\zeta_i \log n, \ldots, c_w(t_w^s) + \gamma\zeta_i \log n]$ in the first place.

In summary, we have that in time slot $t_w^s + 1$, every node $u \in \Gamma(w) \cap \mathcal{A}_i$ has a correct local copy $d_u(w)$ of c_w, and

$$|c_u(t_w^s + 1) - c_w(t_w^s + 1)| > \gamma\zeta_i \log n.$$

Because the counter of every neighbor in $\mathcal{A}_i$ thus differs by at least $\gamma\zeta_i \log n$ from c_w, none of these nodes can cause w to reset its counter in Line 18 of the algorithm. Node w can thus increment its counter in each time slot and hence, all nodes $u \in \Gamma(w) \cap \mathcal{A}_i$ continue to have a correct local copy of c_w after t_w^s. Consequently, even if a neighboring node u has to reset its counter to $\chi(P_u)$, this cannot cause c_u to come within $\gamma\zeta_i \log n$ of

c_w by the definition of $\chi(P_u)$. Thus, it follows by induction over the subsequent time slots that no node $u \in \mathcal{A}_i$ is able to reset w's counter after its successful transmission at time t_w^s. By Lemma 12.8, we know that for all i, $c_w \geq -2\gamma\kappa_2\Delta\log n - 1$ at time t_w^s. Hence, if w stays in $\mathcal{A}_i$, it requires at most $(2\gamma\kappa_2 + \sigma)\Delta\log n + 1$ time slots in order to reach the threshold $\sigma\Delta\log n$, which enables to enter state $\mathcal{C}_i$. Also, nodes that join $\mathcal{A}_i$ *after* t_w^s do not transmit for at least $\alpha\Delta\log n$ time slots, and because $\alpha > 2\gamma\kappa_2+\sigma+1$, it follows that such nodes cannot interfere with w's incrementing its counter either. Hence, after a successful transmission, there remains only one way to prevent w from incessantly incrementing its counter and entering $\mathcal{C}_i$: if w receives a message $M_{\mathcal{C}}^i$ *before* its counter reaches $\sigma\Delta\log n$.

In summary, we have that after a successful transmission, either w enters $\mathcal{C}_i$ itself within $(2\gamma\kappa_2 + \sigma)\Delta\log n + 1$ time slots or there must exist a neighboring node x of w that joins $\mathcal{C}_i$ earlier (see Figure 12.2). In the first case, v receives a message $M_{\mathcal{C}}^i$ from w within $\gamma\zeta_i\log n$ after w's entering $\mathcal{C}_i$ with probability at least $1 - n^{-5}$ (by Lemma 12.2 if $i > 0$ and by Lemma 12.3 if $i = 0$). In the other case, node x (which, in this case, is not a direct neighbor of v) must be a 2-hop neighbor of v. If v is not covered by x and remains in $\mathcal{A}_i$, at least one node $w_2 \in \Gamma(v) \cap \mathcal{A}_i$ can transmit successfully within $\frac{\sigma}{2}\Delta\log n$ time slots thereafter with high probability (Lemma 12.4), and the argument repeats itself. That is, as long as v is active in $\mathcal{A}_i$, at least one node in v's 2-hop neighborhood enters $\mathcal{C}_i$ per $\frac{\sigma}{2}\Delta\log n+(2\gamma\kappa_2+\sigma)\Delta\log n+1$ time slots with probability $1 - n^{-5}$.

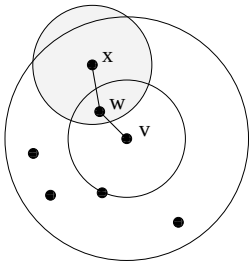

Figure 12.2: When v is active, some neighbor w can transmit successfully within $\frac{\sigma}{2}\Delta\log n$ time slots. This node w can only be blocked from entering $\mathcal{C}_i$ if one of its neighbors x joins $\mathcal{C}_i$ earlier.

It can be seen in Figure 12.2 that the number of times a node $x \in \Gamma_2(v)$ can join $\mathcal{C}_i$ without covering v (and thus forcing v to leave state $\mathcal{A}_i$) is by definition at most κ_2. Finally, once v becomes covered, an additional $\gamma\zeta_i\log n$ time slots in $\mathcal{A}_i$ may be required before, with probability $1 - n^{-5}$, its first neighbor in $\mathcal{C}_i$ sends a message $M_{\mathcal{C}}^i$ to v. As stated at the beginning of the proof, our argument holds under the condition that the set of leaders $\mathcal{C}_0$ forms an independent set which is true with probability $1 - 2n^{-3}$ by Lemma 12.5.

Therefore, with probability P_v, node v spends at most

$$\begin{aligned} T_v^i &\leq \alpha\Delta\log n + \kappa_2\left(\frac{\sigma}{2}\Delta\log n + (2\gamma\kappa_2+\sigma)\Delta\log n + 1\right) + \gamma\zeta_i\log n \\ &\in O(\kappa_2^3\Delta\log n) \end{aligned}$$

time slots in state $\mathcal{A}_i$, where P_v is at least

$$P_v \;\geq\; 1-(\kappa_2\cdot n^{-5}+n^{-5}+2n^{-3}) \;>\; 1-3n^{-3},$$

for large enough n because $\kappa_2 \leq n$ and $\gamma \in O(\kappa_2)$. This concludes the proof. □

Next, we bound the time until a node v in the request state $\mathcal{R}$ receives its intra-cluster color (via a message $M_{\mathcal{C}}^0(w,v,tc_v)$) from its leader w upon which it leaves state $\mathcal{R}$ (cf Line 4 of Algorithm 12.2). Specifically, the following lemma shows that each node v spends at most time $O(\kappa_2\Delta\log n)$ in state $\mathcal{R}$.

Lemma 12.10. *Let $T_v^{\mathcal{R}}$ denote the number of time slots a node v spends in state $\mathcal{R}$. With probability $1-3n^{-3}$ it holds for each $v\in V$ that $T_v^{\mathcal{R}} \leq (\gamma+\beta)\Delta\log n$.*

Proof. The time $T_v^{\mathcal{R}}$ denotes the time between v starting to request an intra-cluster color from its leader $L(v)\in\mathcal{C}_0$ to the time this leader succeeds in assigning the intra-cluster color tc_v to v without collision. Let w be the leader of v, i.e., $w=L(v)$. We divide $T_v^{\mathcal{R}}$ into two parts. First, by Lemma 12.2, v is able to send its request $M_{\mathcal{R}}(v,L(v))$ to w within time $\gamma\Delta\log n$ with probability $1-n^{-5}$. Upon receipt, w queues v's request until it has served all its other, previously received requests. In Line 15 of Algorithm 12.3, w transmits a message $M_{\mathcal{C}}^0$ with probability $1/\kappa_2$ to the currently considered requesting node for $\beta\log n$ time slots, before moving on to the next request, if available. Because $\beta\geq\gamma$, Lemma 12.3 holds for w's response to v with probability $1-n^{-5}$. Because w can have at most Δ requesting nodes in its queue, $T_v^{\mathcal{R}}$ is at most

$$T_v^{\mathcal{R}} \;\leq\; \gamma\Delta\log n + \Delta\cdot\beta\log n = (\gamma+\beta)\Delta\log n$$

for each node $v\in V$ with probability at least $1-2n^{-5}$. As the set $\mathcal{C}_0$ forms an independent set with probability $1-2n^{-3}$, for large enough n the claim holds with probability $1-2n^{-5}-2n^{-3}\geq 1-3n^{-3}$. □

Lemmas 12.9 and 12.10 are the ingredients required to prove the following lemma that bounds the algorithm's running time, i.e., the amount of time every node requires after its wake-up before deciding on a color.

Lemma 12.11. *Every node decides on its color within time $O(\kappa_2^4\Delta\log n)$ after its wake-up with probability $1-4n^{-1}$.*

Proof. Let $T_v^{\mathcal{Y}}$ be the number of time slots a node v spends in state $\mathcal{Y}$. For each node v, we have

$$T_v \;=\; \sum_{i\geq 0} T_v^{\mathcal{A}_i} + T_v^{\mathcal{R}}.$$

Lemma 12.10 bounds $T_v^{\mathcal{R}}$ by $(\gamma+\beta)\Delta\log n$ with probability $1-3n^{-3}$ for each v, and thus with probability $1-3n^{-2}$ for all nodes in V. Moreover, when applying the union bound to the result of Lemma 12.9, it follows that $T_v^{\mathcal{A}_i} \in O(\kappa_2^3\Delta\log n)$ for all v and i with probability $1-3n^{-1}$. Finally, because every node is in at most κ_2+1 different states (due to Corollary 12.7) $\mathcal{A}_i$, it follows that for some constant λ,

$$T_v \;=\; (\kappa_2+1)\cdot\lambda\kappa_2^3\Delta\log n + (\gamma+\beta)\Delta\log n \;\in\; O(\kappa_2^4\Delta\log n)$$

with probability at least $1-4n^{-1}$, for large enough n. □

The only thing remaining is a bound on the number of different colors assigned by the algorithm. For practical purposes, the *locality* of the assignment of colors to nodes plays a crucial role. Generally, the colors assigned to each node should be as "low" as possible. If the vertex coloring in the graph is used for setting up a *time-division scheduling* in a wireless network, for instance, the bandwidth assigned to a node v is often inversely proportional to the value of the *highest color* in its neighborhood. The highest color assigned to a neighbor of a node v by the algorithm in Section 12.1 is dependent only on *local graph properties*. This allows nodes located in low density areas of the network to send more frequently than nodes in dense and congested parts.

Lemma 12.12. *Let $\theta_v := \max_{w\in\Gamma_2(v)}\delta_w$ be the maximum node degree in $\Gamma_2(v)$ and let ϕ_v be the highest color assigned to a node in $\Gamma(v)$. With probability at least $1-2n^{-3}$ the algorithm produces a coloring such that, for all $v\in V$, $\phi_v \leq \kappa_2\cdot\theta_v$.*

Proof. Let $w\in\mathcal{C}_0$ be a leader and let s_w be the number of nodes $v\in\Gamma(w)$ having w as their leader. Leader w assigns unique intra-cluster colors $1,2,\ldots,s_w$ to these nodes. As shown in Corollary 12.7, if the set of leaders forms a correct independent set, a non-leader node v assigned intra-cluster color tc_v ends up selecting a color from the range $tc_v(\kappa_2+1),\ldots,tc_v(\kappa_2+1)+\kappa_2$. Since $s_w\leq\delta_w$ and every node $u\in\Gamma(v)$ is assigned to a leader $w\in\Gamma_2(v)$, the lemma follows. □

Finally, the following main theorem combines the results obtained in Lemmas 12.5, 12.11, and 12.12.

Theorem 12.13. *In any network in which every node has at most κ_2 mutually independent nodes in its two-hop neighborhood, the algorithm produces a correct coloring with at most $\kappa_2\Delta$ colors with probability $1-2n^{-3}$. Furthermore, with probability $1-4n^{-1}$ every node irrevocably decides on its color $O(\kappa_2^4\Delta\log n)$ time slots after its wake-up.*

Chapter 13

Computing an MIS in Radio Networks

In the previous chapter, we have presented an efficient algorithm for computing a coloring in an unstructured radio network. In this chapter, we return to one of the core problems considered in Part I of this thesis, the maximal independent set (MIS) problem. In particular, we seek to establish the complexity of computing an MIS in the unstructured radio network model by providing a probabilistic algorithm for the problem.

In comparison to the basic unstructured radio network model introduced in Chapter 11, this section deals with a slightly adapted variant of the model. Specifically, we change the model in two regards. First, we assume nodes to be located in a metric space with low doubling dimension and study the corresponding unit ball graph (cf Chapter 8). For simplicity, we actually present the result as well as its analysis for the case of unit disk graphs only. Secondly, in addition to the restrictions imposed by the unstructured radio network model, we limit the amount of information stored on each node at any given time to be strictly limited to $O(\log n)$ bits in this section. That is, no node is capable of storing more than a constant number of integer values, counters, or neighbor identifiers. Notice that this second restriction adds to the harshness of the model and, for instance, the coloring algorithm presented in Chapter 12 is not implementable in this model as it requires nodes to store the competitor list, a list of neighboring counter values.

As already pointed out in Section 11.2, there exists only a small body of related work that deals with the construction of network coordination structures in radio networks models. The two works most relevant to the MIS algorithm presented in this chapter are [86] and [138]. The authors of [138] have studied the wake-up problem in an unstructured radio network model in the single-hop case in which all nodes are within mutual communication range. The recent lower bound of [86] shows that even in a single-hop environment, the number of time slots required until, with high probability, at

least one node can transmit without collision is at least $\Omega(\log^2 n)$, thereby also placing a lower bound on the time complexity of any MIS algorithm in this model.

13.1 Algorithm

This section presents the MIS algorithm. The general idea of the algorithm is to combine methods used in the coloring algorithm of Chapter 12 with an exponential increase of transmission probabilities similar to the one studied in the single-hop case in [138]. In more detail, the procedure is described in Algorithm 13.1. Every time slot corresponds to one iteration of the main loop. The *Receive Triggers* are executed immediately after the receipt of a message, regardless of the current state of the algorithm. In accordance to the model, however, a node receives a message only if it does not send a message itself in the same time slot.

At any time during the execution of Algorithm 13.1, a node can be in one of five states. Upon waking up, a node is in the *waiting state* $\mathcal{W}$ in which it only listens. If a node does not become covered by an MIS node in this state already, it eventually becomes *active*. Active nodes are in state $\mathcal{A}$. An active node v tries to join the MIS by increasing its probability p_v of becoming a *candidate*. Eventually, some active nodes will become candidates by entering state $\mathcal{C}$, whereas others will restart the algorithm, returning to the initial waiting state $\mathcal{W}$. Finally, the MIS nodes are elected from among the candidates. Nodes that have decided to be an MIS node end up in state $\mathcal{L}$, nodes that are covered become *slaves* and enter state $\mathcal{S}$. Throughout the paper, we use the expression $\mathcal{W}$ to denote both the *state* in which the algorithm is currently in, as well as the subset of nodes $v \in V$ that are currently in the state $\mathcal{W}$. The same holds for all other states/sets. The implementation of the different states is described in more detail in the sequel. In the waiting state $\mathcal{W}$, a node listens for messages and increases the counter *step* in each time slot. The purpose of state $\mathcal{W}$ is that newly awakening or restarting nodes should not interfere with nodes that are actively competing for becoming an MIS node.

Once the *step* counter of a node $v \in \mathcal{W}$ reaches the threshold $4\mu\delta \log^2 n$ (Line 3), it proceeds to the *active state* $\mathcal{A}$. Every active node has a sending probability p_v which is the probability that it transmits a message $M_{\mathcal{A}}$ and becomes a *candidate* in a given time slot (Lines 9-11). Starting from a small initial probability p_v, a node $v \in \mathcal{A}$ doubles p_v every $\lambda \log n$ time slots, thereby exponentially increasing its chance to become a candidate (Lines 6 and 7). If, however, an active node $v \in \mathcal{A}$ receives a message $M_{\mathcal{A}}$ from another active node, it returns to the start of the algorithm, i.e., it sets its state to $\mathcal{W}$ and resets *step* to 0 (Receive Trigger 1). Such nodes may again try to become a candidate subsequently. State $\mathcal{A}$ is designed to bound the number of candidates simultaneously being in state $\mathcal{C}$ in a certain area of the graph. This enables a quick election of MIS nodes among the limited number of candidates. In other words, the purpose of state $\mathcal{A}$ is a first rough selection on the way towards picking MIS nodes.

$step := c_v := 0$; $state := \mathcal{W}$;
upon wake-up do:
1: **loop**
2: **case** *state* **do**
3: $\mathcal{W}$: **if** $step \geq 4\mu\delta \log^2 n$ **then**
4: $state := \mathcal{A}$; $step := 0$; $p_v := \frac{2^{-\alpha-1}}{n}$;
5: **end if**
6: $\mathcal{A}$: **if** $step \geq \lambda \log n$ **then**
7: $p_v := 2p_v$; $step := 0$;
8: **else**
9: $s := \begin{cases} 1 & \text{with probability } p_v \\ 0 & \text{with probability } 1 - p_v \end{cases}$
10: **if** $s = 1$ **then**
11: **transmit** $M_{\mathcal{A}}$; $state := \mathcal{C}$; $step := 0$;
12: **end if**
13: **end if**
14: $\mathcal{C}$: **if** $step > \beta \log n$ **then**
15: $c_v := c_v + 1$;
16: $s := \begin{cases} 1 & \text{with prob. } q_C = \frac{\tau}{2^\alpha \log n} \\ 0 & \text{with prob. } 1 - q_C \end{cases}$
17: **if** $c_v \geq \delta \log^2 n$ **then**
18: $state := \mathcal{L}$ $\{* v \text{ joins MIS } *\}$
19: **else if** $s = 1$ **then**
20: **transmit** $M_{\mathcal{C}}(c_v)$;
21: **end if**
22: **end if**
23: $\mathcal{L}$: **transmit** $M_{\mathcal{L}}$ with probability $q_L = 2^{-\alpha}$
24: **end case**
25: $step := step + 1$;
26: **end loop**

Receive Triggers (only when not sending):
1: upon receiving $M_{\mathcal{A}}$ **do:**
if $state = \mathcal{W}$ **or** $state = \mathcal{A}$ **then**
$state := \mathcal{W}$; $step := 0$
end if
2: upon receiving $M_{\mathcal{C}}(c_w)$ **do:**
$\Delta c := |c_w - c_v|$;
if $state = \mathcal{C}$ **and** $\Delta c \leq \beta \log n$ **then**
$c_v := 0$; $step := 0$;
end if
3: upon receiving $M_{\mathcal{L}}$ **do:**
$state := \mathcal{S}$; **stop()**; $\{* v \text{ becomes slave } *\}$

Algorithm 13.1: MIS-Algorithm (Code of node v)

Having bounded the number of *candidates*, it remains to select MIS nodes from among the candidates. Basically, the idea is to use a method similar to the one employed in the coloring algorithm presented in Chapter 12 restricted to the selected candidates. That is, all candidates compete for joining the MIS by increasing a counter c_v and as soon as this counter reaches a certain threshold, they are eligible to enter the MIS. However, due to the inability to store a list of neighboring counter values in the adapted model, the method employed in the coloring algorithm using competitor lists must be adjusted. In particular, instead of maintaining the counters of all neighboring nodes, each node that resets its counter must wait for some time before restarting to increment its counter. This additional waiting period gives a node that was able to transmit *successfully* enough time to increase its counter to a high enough value, so that this candidate cannot be reset anymore by any other candidate.[1] From the point of view of the analysis, the additional waiting period creates the problem that there may be time slots in which there is *no progress* at all, i.e., no node increases its counter. The analysis in Section 13.2 shows, however, that the probability of there being a large number of such bad time slots is small.

More precisely, the approach can be described as follows: Neighboring candidates compete with each other such that no two neighboring nodes join state $\mathcal{L}$. They do so by means of a counter variable c_v. Intuitively, the current value of the c_v describes a node's progress towards joining the MIS. In each time slot with $step > \beta \log n$, a candidate v increases its c_v value and sends a message $M_{\mathcal{C}}(c_v)$ containing its current c_v value with probability q_C. When receiving a message $M_{\mathcal{A}}(c_w)$ from another candidate, the receiver compares the sender's c_w to its own. If the two values are within $\beta \log n$ of each other, the receiver resets its own c_v and *step* (Receive Trigger 2), and thus, does not increase c_v or transmit for the next $\beta \log n$ time slots. The idea is that a candidate v resets its c_v and *step* if the progress of v and w are too close to one another. Note that if w transmits $M_{\mathcal{C}}(c_v)$ successfully to all its neighbors (without collision) then its counter cannot be reset anymore by another candidate, because at most $\beta \log n$ time slots later, it differs from any other counter in its neighborhood by more than $\beta \log n$. As shows in Section 13.2, this method of comparing counters prevents two neighboring nodes from joining the MIS shortly in succession and consequently ensures the correctness of the resulting MIS. On the other hand, it also allows some nodes to make fast progress in all parts of the network graph. Once a candidate's counter c_v reaches the threshold $\delta \log^2 n$, it becomes an MIS node and enters its final state $\mathcal{L}$ (Line 17). MIS nodes continue to transmit messages $M_{\mathcal{L}}$ with a probability q_L in order to inform their neighbors that they are covered. Regardless of its current state, if a node receives a messages $M_{\mathcal{L}}$ during the algorithm (Receive Trigger 3), it decides to *become a slave*.

The constants μ and α are defined as $\mu = 19$ and $\alpha = 6.4$, respectively. The other constant parameters can again be chosen to fine-tune the trade-off between running time and the probability of a correct execution. In order to

[1] Like in the previous chapter, we say that a node can transmit *successfully* if it transmits and all its neighbors receive the message.

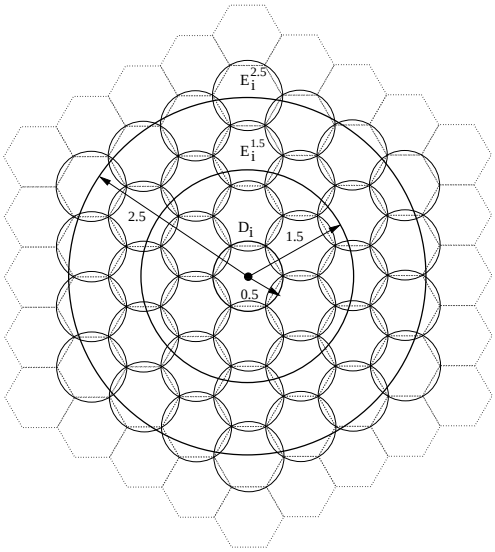

Figure 13.1: Circles D_i, $E_i^{1.5}$, and $E_i^{2.5}$

obtain the high probability results in Section 13.2, the constants can be set as $\lambda = 3 \cdot 2^{\alpha+2} 4^{\frac{9/4+3\mu}{2^\alpha}}$,

$$\beta = \frac{8 \cdot 2^\alpha}{\tau} \cdot 4^{\frac{6\mu}{2^\alpha}} \quad \text{and} \quad \delta = 6\beta \cdot \left(\frac{1}{e} \left(1 - \frac{\tau}{2^{\alpha+2}} \right) \right)^{-\frac{\mu}{2^\alpha}} \cdot 4^{\frac{2\mu}{2^\alpha}}$$

and $\tau = 9000^{-1}$.

Finally, note that like in Chapter 12, different nodes may be in different states at the same time and no node has a-priori knowledge about the current states of its (potential) neighbors.

13.2 Analysis

This section proves that Algorithm 13.1 computes a correct MIS in time $O(\log^2 n)$ with high probability. We make use of an imaginary covering of the plane by disks D_i of radius 1/2 as shown in Figure 13.1. By E_i^r, we denote the disk with radius r centered at the center of D_i. Observe that all nodes within a disk D_i can hear each other. On the other hand, a node outside $E_i^{1.5}$ cannot cause a collision at a node $v \in D_i$. The following geometric facts can be proven by standard area arguments.

Fact 13.1. *Disks $E_i^{1.5}$ and $E_i^{2.5}$ can be fully covered by μ and 2μ smaller disks D_i, respectively, where $\mu = 19$. Also, the number of independent nodes in $E_i^{1.5}$, and $E_i^{2.5}$ is at most μ and 2μ, respectively.*

The main difficulty of the analysis is that nodes can unintentionally interfere with neighbors in different states. For instance, an active node v may cause a collision at a candidate node w in $\mathcal{C}$. This results in w not receiving a message $M_{\mathcal{C}}$ or $M_{\mathcal{L}}$ from a neighboring node, potentially causing a violation of the MIS condition. Or, a collision caused by an MIS message $M_{\mathcal{L}}$ may cause that a node in $\mathcal{A}$ does not receive a message $M_{\mathcal{A}}$, which could lead to too many candidates.

Throughout the proof, $\mathcal{A}_i$ and $\mathcal{A}_i^r$ denote the set of active nodes in D_i and E_i^r, respectively. $\mathcal{W}_i$, $\mathcal{C}_i$, and the other sets are defined analogously. We begin with a definition and a simple observation that follows directly from Algorithm 13.1.

Definition 13.1. *Let t be a time slot in which a message $M_{\mathcal{A}}$ is transmitted by a node $v \in \mathcal{A}_i$ and received without collision by all nodes $w \in D_i \setminus \{v\}$. We call t a* clearance *of D_i. Two subsequent clearances are* independent *if they are not caused by the same node.*

Lemma 13.1. *Consider a disk D_i. After a clearance, no node $v \in D_i$ is in state $\mathcal{A}$ for the next $4\mu\delta \log^2 n$ time slots. Consequently, two independent clearances in the same disk D_i must be at least $4\mu\delta \log^2 n$ time slots apart.*

A critical ingredient of the analysis is to bootstrap the argument. In this section, we show that with high probability the algorithm maintains three properties (probabilistic invariants) throughout its execution. The proof then works in the form of an induction over all three properties. The first property states that the sum of sending probabilities by active nodes does never exceed a certain constant. This helps to bound the "noise" caused by such nodes when analyzing other aspects of the algorithm.

Property 13.1 (P1). *For all disks D_i and at any time slot t throughout the execution of the algorithm, it holds that $\sum_{v \in \mathcal{A}_i} p_v(t) \leq 2^{-\alpha}$.*

The second and third properties state that the number of simultaneous candidates is bounded and that $\mathcal{L}$ forms a correct independent set, respectively.

Property 13.2 (P2). *For all disks D_i and at any time slot t throughout the execution of Algorithm 13.1, it holds that $|\mathcal{C}_i| \leq \tau^{-1} \log n$.*

Property 13.3 (P3). *Throughout the execution of the algorithm, the set $\mathcal{L}$ forms a correct independent set.*

The first technical lemma shows that MIS nodes are capable of quickly informing their neighbors that they are covered. This is necessary to ensure the independence of the resulting set $\mathcal{L}$. For now, we can formalize this intuition only under the assumption that all three Properties hold.

Lemma 13.2. *Assume Properties 13.1, 13.2, and 13.3 hold. With probability $1 - n^{-3}$, every node $v \in V$ joins $\mathcal{S}$ and terminates the algorithm by the time $t_v + \beta \log n$, where t_v is the first time slot in which v becomes covered by an MIS node $w \in \mathcal{L} \cap \Gamma(v)$.*

Proof. Consider a node $v \in D_i$ and let t_v be the time slot defined in the lemma. The probability P_1 that in an arbitrary time slot $t \in [t_v + 1, \ldots, t_v + \beta \log n]$ MIS node $w \in \mathcal{L} \cap \Gamma(v)$ transmits and no other node in $\Gamma(v)$ sends (i.e., that v receives w's message $M_{\mathcal{L}}$) is at least

$$P_1 \;\geq\; q_L \prod_{u \in \Gamma(v)} (1 - p_u) \underset{\text{Fact 2.1}}{\geq} \frac{1}{2^\alpha} \left(\frac{1}{4}\right)^{\sum_{u \in \Gamma(v)} p_u}.$$

To bound $\sum_{u \in \Gamma(v)} p_u$, we make use of the assumption that the three Properties hold and that nodes in $\mathcal{W} \cup \mathcal{S}$ do not transmit. It follows that the sum of sending probabilities in $\Gamma(v)$ is upper bounded by

$$\begin{aligned}
\sum_{u \in \Gamma(v)} p_u &= \sum_{u \in \mathcal{A} \cap \Gamma(v)} p_u + \sum_{u \in \mathcal{C} \cap \Gamma(v)} p_u + \sum_{u \in \mathcal{L} \cap \Gamma(v)} p_u \\
&\leq \sum_{D_j \in E_i^{1.5}} \left(\sum_{u \in \mathcal{A}_j} p_u + \sum_{u \in \mathcal{C}_j} q_C + \sum_{u \in \mathcal{L}_j} q_L \right) \\
&\leq \sum_{D_j \in E_i^{1.5}} \left(\frac{1}{2^\alpha} + \tau^{-1} \log n \cdot \frac{\tau}{2^\alpha \log n} + \frac{1}{2^\alpha} \right) \\
&\leq \mu \left(\frac{1}{2^\alpha} + \frac{\log n}{2^\alpha \log n} + \frac{1}{2^\alpha} \right) \;=\; \frac{3\mu}{2^\alpha},
\end{aligned}$$

where the second inequality is derived by replacing $q_C = \frac{\tau}{2^\alpha \log n}$ and $q_L = 2^{-\alpha}$ as defined in Algorithm 13.1. The third inequality follows from Fact 13.1. Plugging this in the expression for P_1 yields $P_1 \geq 2^{-\alpha} (1/4)^{\frac{3\mu}{2^\alpha}}$. The probability P_{no} that none of the $\beta \log n$ time slots in the interval $[t_v+1, \ldots, t_v+\beta \log n]$ is successful is at most

$$P_{no} \;\leq\; \left(1 - \frac{1}{2^\alpha} \left(\frac{1}{4}\right)^{\frac{3\mu}{2^\alpha}}\right)^{\beta \log n} \;\leq\; e^{-\beta \log n \cdot \frac{1}{2^\alpha} (1/4)^{\frac{3\mu}{2^\alpha}}}$$

which is, by the definitions of α, μ, and β, $P_{no} \in O(n^{-4})$. Finally, the argument is concluded by the observation that every node needs to be informed about its being covered at most once. That is, the claim holds for all nodes $v \in V$ with probability $1 - O(n^{-3})$. □

One difficulty when proving that the three Properties hold will be that the results depend on arguments of the following kind: *Before* a particular Property can be violated, there must exist some time-interval in which certain

critical conditions hold. We can prove that after any interval exhibiting these conditions, the Property will *not* be violated with high probability. However, such an argument is useless if there can be infinitely many time-intervals with these critical conditions. To bound the number of critical time-intervals, the next lemma lower bounds the *progress* made by the algorithm in case all three properties hold.

Lemma 13.3. *Assume Properties 13.1, 13.2, and 13.3 hold. Let t_c be a time slot in which an uncovered node $v \in D_i$ is a candidate, i.e., $v \in \mathcal{C}$. In the interval $[t_c - \beta \log n, \ldots, t_c + 2\delta \log^2 n]$, a new MIS node emerges in $E_i^{2.5}$ with probability $1 - O(n^{-3})$.*

Proof. We first show that unless all candidates in D_i become covered by an MIS node (in which case the lemma clearly holds), there is at least one candidate that can send successfully in the interval $\mathcal{I}_1 = [t_c + 1, \ldots, t_c + \delta \log^2 n]$. Clearly, if a node $w \in \Gamma(v) \cap \mathcal{C}$ sends and no other node in $E_i^{2.5}$ sends, then w sends successfully.

We call a candidate *active* if its *step* is larger than $\beta \log n$, that is, if it is not waiting during the empty $\beta \log n$ time slots following a *counter* reset. Active candidates transmit with probability q_C. We name a time slot *good* for a candidate v if there is at least one active candidate in $\Gamma(v)$, possibly v itself. Otherwise, the time slot is *bad*. In the proof, we independently bound the number of good and bad time slots before a candidate $w \in \Gamma(v) \cap \mathcal{C}$ manages to send successfully.

For the good time slots, we use the fact that in every good time slot, there is at least one candidate that has a non-zero sending probability. The probability P_1 that there is a candidate $w \in \Gamma(v) \cap \mathcal{C}$ sending successfully in a good time slot $t \in \mathcal{I}_1$ is at least

$$\begin{aligned} P_1 &\geq \sum_{w \in \Gamma(v) \cap \mathcal{C}} \left(p_w \cdot \prod_{u \in E_i^{2.5} \setminus \{w\}} (1 - p_u) \right) \\ &\geq \sum_{w \in \Gamma(v) \cap \mathcal{C}} p_w \cdot \prod_{u \in E_i^{2.5}} (1 - p_u) \\ &\underset{\text{Fact 2.1}}{\geq} \sum_{w \in \Gamma(v) \cap \mathcal{C}} p_w \cdot (1/4)^{\sum_{u \in E_i^{2.5}} p_u}. \end{aligned}$$

Similar to the proof of Lemma 13.2, we can bound the sum in the exponent as

$$\begin{aligned} \sum_{u \in E_i^{2.5}} p_u &= \sum_{u \in \mathcal{A}_i^{2.5}} p_u + \sum_{u \in \mathcal{C}_i^{2.5}} p_u + \sum_{u \in \mathcal{L}_i^{2.5}} p_u \\ &\leq \sum_{D_j \in E_i^{2.5}} \left(\frac{1}{2^\alpha} + \frac{\tau \log n}{2^\alpha \tau \log n} + \frac{1}{2^\alpha} \right) \leq \frac{6\mu}{2^\alpha}. \end{aligned}$$

Unless all candidates receive a message $M_{\mathcal{L}}$ and join the set $\mathcal{S}$ in $\mathcal{I}_1$, the fact that the time slot is good implies that there is at least one active candidate

node in $\Gamma(v) \cap \mathcal{C}$ and therefore $\sum_{w \in \Gamma(v) \cap \mathcal{C}} p_w \geq q_C$. Plugging these results into the above expression for P_1 yields $P_1 \geq \frac{\tau}{(2^\alpha \log n)} (1/4)^{\frac{6\mu}{2^\alpha}}$ and the probability P_{no} that during at least $\frac{\delta}{2} \log^2 n$ good time slots no candidate $w \in \Gamma(v) \cap \mathcal{C}$ transmits successfully is at most

$$\begin{aligned} P_{no} &\leq \left(1 - \frac{\tau}{(2^\alpha \log n)} \cdot \left(\frac{1}{4}\right)^{\frac{6\mu}{2^\alpha}}\right)^{\frac{\delta}{2} \log^2 n} \\ &\underset{\text{Fact 2.2}}{\leq} e^{-\frac{\delta}{2} \log n \cdot \frac{\tau}{2^\alpha} (1/4)^{\frac{6\mu}{2^\alpha}}} \in O(n^{-3}). \end{aligned}$$

Therefore, with probability $1 - O(n^{-3})$, the number of good time slots before a candidate $w \in \Gamma(v) \cap \mathcal{C}$ transmits successfully is bounded by $\frac{\delta}{2} \log^2 n$.

Next, we bound the number of bad time slots before a candidate $w \in \Gamma(v) \cap \mathcal{C}$ transmits successfully. The idea is to show that there cannot be too many bad time slots without there being also some good time slots.

Let v be an arbitrary candidate. For a given time slot, let *Suc* be the event that an active candidate in v's neighborhood transmits successfully and let *Send* be the event that there is an active candidate in v's neighborhood that transmits. The conditional probability that if a candidate transmits, it does so successfully is given by

$$P[Suc \mid Send] = \frac{P[Suc \cap Send]}{P[Send]} = \frac{P[Suc]}{P[Send]}$$

The probability that a candidate in $\Gamma(v) \cap \mathcal{C}$ transmits successfully in a given time slot t is lower bounded by

$$P[Suc] \geq h(t) \cdot q_C \cdot (1 - q_C)^{h(t)-1} \cdot \prod_{w \in \Gamma(v) \cap (\mathcal{A} \cup \mathcal{L})} (1 - p_w),$$

where $h(t)$ denotes the number of active candidates in $\Gamma(v)$ during time slot t. By Fact 2.1, it holds that

$$\prod_{w \in \Gamma(v) \cap (\mathcal{A} \cup \mathcal{L})} (1 - p_w) \geq \left(\frac{1}{4}\right)^{\sum_{w \in \Gamma(v) \cap (\mathcal{A} \cup \mathcal{L})} p_w}.$$

Similarly, as in the proof of Lemma 13.2 we obtain

$$\begin{aligned} \sum_{w \in \Gamma(v) \cap (\mathcal{A} \cup \mathcal{L})} p_w &= \sum_{u \in \mathcal{A} \cap \Gamma(v)} p_u + \sum_{u \in \mathcal{L} \cap \Gamma(v)} p_u \\ &\leq \sum_{D_j \in E_i^{1.5}} \left(\frac{1}{2^\alpha} + \frac{1}{2^\alpha}\right) \leq \frac{2\mu}{2^\alpha}. \end{aligned}$$

Furthermore, under the assumption that Property 13.2 holds, $h(t) \leq \frac{\mu}{\tau} \log n$ and thus

$$\begin{aligned}(1-q_C)^{h(t)-1} &\geq \left(1-\frac{\tau}{2^\alpha \log n}\right)^{\frac{\mu}{\tau}\log n} = \left(\left(1-\frac{\tau}{2^\alpha \log n}\right)^{\frac{2^\alpha \log n}{\tau}}\right)^{\frac{\mu}{2^\alpha}} \\ &\underset{\text{Fact 2.2}}{\geq} \left(\frac{1}{e}\left(1-\frac{\tau}{2^\alpha \log n}\right)\right)^{\frac{\mu}{2^\alpha}} \geq \left(\frac{1}{e}\left(1-\frac{\tau}{2^{\alpha+2}}\right)\right)^{\frac{\mu}{2^\alpha}}\end{aligned}$$

for $n \geq 16$. Putting things together, we obtain

$$\begin{aligned}P[Suc \mid Send] &\geq (1-q_C)^{h(t)-1} \cdot \prod_{w \in \Gamma(v) \cap (\mathcal{A} \cup \mathcal{L})} (1-p_w) \\ &\geq \left(\frac{1}{e}\left(1-\frac{\tau}{2^{\alpha+2}}\right)\right)^{\frac{\mu}{2^\alpha}} \cdot \left(\frac{1}{4}\right)^{\frac{2\mu}{2^\alpha}}.\end{aligned}$$

Hence, $P[Suc \mid Send] \in \Omega(1)$. This means that whenever a candidate in v's neighborhood transmits, it transmits successfully with constant probability. Or turning this around, whenever node v receives a message and resets its counter, the transmitting node has transmitted successfully with constant probability.

Therefore, the probability that there are at least $\frac{\delta}{2\beta} \log n$ time slots with non-successful transmissions that can reset v's counter before at least one candidate in $\Gamma(v)$ sends successfully is at most

$$\begin{aligned}&\left(1-\left(\frac{1}{e}\left(1-\frac{\tau}{2^{\alpha+2}}\right)\right)^{\frac{\mu}{2^\alpha}} \cdot \left(\frac{1}{4}\right)^{\frac{2\mu}{2^\alpha}}\right)^{\frac{\delta}{2\beta}\log n} \\ \underset{\text{Fact 2.2}}{\leq}\ & e^{-\frac{\delta}{2\beta}\log n\left(\frac{1}{e}\left(1-\frac{\tau}{2^{\alpha+2}}\right)\right)^{\frac{\mu}{2^\alpha}} \cdot \left(\frac{1}{4}\right)^{\frac{2\mu}{2^\alpha}}} \in O(n^{-3}).\end{aligned}$$

Since every such reset can incur at most $\beta \log n$ bad time slots, the number of bad time slots before a candidate node in $\Gamma(v)$ transmits successfully is at most $\frac{\delta}{2\beta} \log n \cdot \beta \log n = \frac{\delta}{2} \log^2 n$ with probability $1 - O(n^{-3})$.

Altogether, the number of good or bad time slots before a candidate node in $\Gamma(v)$ transmits successfully is bounded by $\delta \log^2 n$ with probability $1 - O(n^{-3})$.

Finally, after a candidate node $w \in \Gamma(v)$ transmits successfully, no other neighboring candidate can reduce w's counter anymore. While this is the same argument as used in the coloring algorithm presented in Chapter 12, the reason is actually a different one. In the coloring algorithm, every node keeps a list of neighboring counters and—when transmitting or resetting—increases its counter to a carefully selected value that lies outside the critical range of any neighbor in this list. In Algorithm 13.1 on the other hand, this method cannot be implemented as it requires too much memory. Instead,

counters are always reset to 0, but nodes do not increase their counters for $\beta \log n$ time slots after a reset. Because of this, it is guaranteed that the counters of neighboring candidates never come within the critical range of a successful sender w.

That is, the only possible way for w to be stopped from joining $\mathcal{L}$ is if it receives a message $M_{\mathcal{L}}$ from a node $x \in \Gamma(w)$ that has joined $\mathcal{L}$ *before* w. Thus, w or one of its neighbors will join the MIS after $2\delta \log^2 n$ with probability $1 - O(n^{-3})$. □

Having proven Lemma 13.3 allows us to derive that once a node becomes a candidate, it either joins the MIS or becomes covered shortly thereafter, if all three Properties hold.

Lemma 13.4. *Assume Properties 13.1, 13.2, and 13.3 hold. Let t be an arbitrary time slot. Every node v that is a candidate at time t, i.e., $v \in \mathcal{C}$, will either have joined $\mathcal{L}$ or be covered by time $t + 4\mu\delta \log^2 n$ with probability $1 - O(n^{-2})$.*

Proof. Because all three Properties are assumed to be true, we can prove the claim by repeatedly applying Lemma 13.3. Consider a node $v \in D_i$ that is a candidate at time t. We know by Lemma 13.3 that there is at least one MIS node in $E_i^{2.5}$ by $t + 2\delta \log^2 n$ with probability $1 - O(n^{-3})$. If v is covered by this new MIS node, the lemma holds. If not, v is still a candidate and hence, again by Lemma 13.3, there is an MIS node emerging in the interval $[t + 2\delta \log^2 n - \beta \log n, \ldots, t + 4\delta \log^2 n]$ with high probability. Thus applying Lemma 13.3 repeatedly yields that an MIS node emerges in the interval $[t + 2j\delta \log^2 n - \beta \log n, \ldots, t + 2(j+1)\delta \log^2 n]$ with probability $1 - O(n^{-3})$ for every $j \geq 0$, as long as v is uncovered. Since every emerging MIS node can cover only two "adjacent" intervals, it holds with probability $(1 - O(n^{-3}))^j$ that there are at least $\lceil j/2 \rceil$ new MIS nodes emerging in $E_i^{2.5}$ if v is still uncovered by time $t + 2j\delta \log^2 n$. Due to Property 13.3, we assume the set $\mathcal{L}$ to be a correct independent set which means that there can be at most 2μ MIS nodes in $E_i^{2.5}$. Hence, it follows that with probability $(1 - O(n^{-3}))^{2\mu} \in 1 - O(n^{-3})$, v is covered by the time $t + 4\mu\delta \log^2 n$. Since there are at most n candidates, the Lemma holds for all nodes with probability $(1 - O(n^{-3}))^n \in 1 - O(n^{-2})$. □

We now return to the notion of a *clearance* which will be crucial in proving the validity of Properties 13.1 and 13.2. In particular, we use the two previous lemmas to bound the number of clearances that can occur in a disk D_i during the execution of the algorithm.

Lemma 13.5. *Assume Properties 13.1, 13.2, and 13.3 hold. For all disks D_i, there are no more than μ independent clearances in D_i with probability $1 - O(n^{-2})$.*

Proof. By Lemma 13.1, there can be at most one independent clearance every $4\mu\delta \log^2 n$ time slots in a disk D_i. Let t_c be a clearance of D_i. By definition, exactly one active node $v \in \mathcal{A}_i$ transmits successfully in time

slot t_c. By definition of Algorithm 13.1, this node becomes a candidate and by Lemma 13.3, there is a node $w \in E_i^{2.5}$ that joins the MIS in the interval $[t_c - \beta \log n, \ldots, t_c + 2\delta \log^2 n]$ with probability $1 - O(n^{-3})$. Hence, after μ independent clearances, at least μ MIS nodes have emerged in $E_i^{2.5}$ with probability $(1 - O(n^{-3}))^{\mu} \in 1 - O(n^{-3})$. By Fact 13.1 and under the assumption of Property 13.3, these μ MIS nodes entirely cover $E_i^{2.5}$ and all nodes located therein. Moreover, it follows from Lemma 13.2 that every such node receives a message $M_{\mathcal{L}}$ $\beta \log n$ time slots after its becoming covered with probability $1 - n^{-3}$. That is, no node will be in the active state $\mathcal{A}$ once all of $E_i^{2.5}$ is covered. With probability $1 - O(n^{-3})$, this is the case after μ clearances. □

All previous lemmas are derived under the condition that Properties 13.1, 13.2, and 13.3 hold. At the beginning of the algorithm—when the first node wakes up—, all three properties are clearly satisfied. That is, if one of the three properties is to be violated, there must be a time slot t, in which at least one of the properties is violated, while for all $t' < t$, all properties hold. In the following sequence of Theorems 13.6, 13.7, and 13.9, we show that with high probability none of the three properties is among the *first* to be violated.

Theorem 13.6. *Assume Property 13.1 is among the first properties to be violated and let t_1 be the first time slot in which the violation occurs. The probability that there exists such a time slot t_1 during the execution of Algorithm 13.1 is at most $P_{fail} \in O(n^{-1})$.*

Proof. If t_1 is the first time slot in which the violation occurs in a disk D_i, it holds $\sum_{v \in \mathcal{A}_i} p_v(t_1 - 1) \leq 2^{-\alpha}$ and $\sum_{v \in \mathcal{A}_i} p_v(t_1) > 2^{-\alpha}$. Consider the interval $\mathcal{I} = [t_1 - \lambda \log n, \ldots, t_1 - 1]$. By the definition of Algorithm 13.1 (Lines 6 and 7), every active node $v \in \mathcal{A}_i$ doubles its sending probability p_v exactly once during this interval $\mathcal{I}$. Additionally, new nodes that were previously in state $\mathcal{W}_i$ may join the set $\mathcal{A}_i$ during $\mathcal{I}$, but these nodes' combined sending probability is at most $n \cdot \frac{2^{-\alpha-1}}{n} = 2^{-\alpha-1}$, according to the definition of a node's initial sending probability. That is, the sum of sending probabilities at time $t_1 - \lambda \log n$ is at least

$$\sum_{v \in \mathcal{A}_i} p_v(t_1 - \lambda \log n) \;\geq\; \frac{1}{2}(2^{-\alpha} - 2^{-\alpha-1}) \;=\; 2^{-\alpha-2}.$$

Consequently, *if* Property 13.1 is violated, there must be an interval $\mathcal{I}$ preceding the violation during which the sum of the sending probabilities is in the range

$$2^{-\alpha-2} \;\leq\; \sum_{v \in \mathcal{A}_i} p_v(t) \;\leq\; 2^{-\alpha} \quad \forall t \in \mathcal{I}. \tag{13.1}$$

In all neighboring disks $D_j \in E_i^{1.5}$, the sum of sending probabilities is

$$0 \;\leq\; \sum_{v \in \mathcal{A}_j} p_v(t) \;\leq\; 2^{-\alpha} \quad \forall t \in \mathcal{I} \tag{13.2}$$

because t_1 is the first time slot violating Property 13.1.

The proof is continued by showing that with high probability, a clearance occurs in the interval $\mathcal{I}$. For that purpose, let P_{no} be the probability that in a given time slot $t \in \mathcal{I}$ no node in $E_i^{1.5} \setminus D_i$ transmits. By P_{one} we denote the probability that exactly one node in D_i transmits in t. The probability P_{clear} of a clearance at time t is $P_{clear} = P_{one} \cdot P_{no}$. Using Fact 2.1, the probabilities P_{one} and P_{no} can be bounded as follows:

$$\begin{aligned} P_{one} &= \sum_{v \in \mathcal{A}_i} \left(p_v \prod_{w \in D_i \setminus \{v\}} (1 - p_w) \right) \geq \sum_{v \in \mathcal{A}_i} p_v \prod_{w \in D_i} (1 - p_w) \\ &\underset{\text{Fact 2.1}}{\geq} \sum_{v \in \mathcal{A}_i} p_v (1/4)^{\sum_{w \in D_i} p_w} \geq \sum_{v \in \mathcal{A}_i} p_v (1/4)^{\sum_{w \in \mathcal{A}_i} p_w + \frac{1}{2^{\alpha-1}}}, \end{aligned}$$

where the last inequality holds because of $\sum_{w \in \mathcal{C}_i} p_w \leq 1/2^\alpha$ and $\sum_{w \in \mathcal{L}_i} p_w \leq 1/2^\alpha$ under the assumption that Properties 13.2 and 13.3 hold, which is the case during $\mathcal{I}$ because t_1 is the first time slot in which a property is violated. Further,

$$\begin{aligned} P_{no} &= \prod_{v \in E_i^{1.5}} (1 - p_v) \geq \prod_{D_j \in E_i^{1.5}} \prod_{v \in D_j} (1 - p_v) \\ &\geq \prod_{D_j \in E_i^{1.5}} (1/4)^{\sum_{v \in D_j} p_v} \\ &\geq \left[(1/4)^{\sum_{v \in \mathcal{A}_j} p_v + \frac{1}{2^{\alpha-1}}} \right]^\mu \geq (1/4)^{\frac{3\mu}{2^\alpha}} \end{aligned}$$

where the last step follows from (13.2). The probability of $t \in \mathcal{I}$ being a clearance is therefore at least

$$P_{clear} \geq \sum_{v \in \mathcal{A}_i} p_v (1/4)^{\sum_{w \in \mathcal{A}_i} p_w + \frac{1}{2^{\alpha-1}}} \cdot (1/4)^{\frac{3\mu}{2^\alpha}}.$$

For $x \in [2^{-\alpha-2}, \ldots, 2^{-\alpha}]$, the function $x(1/4)^{x + \frac{1}{2^{\alpha-1}}}$ is minimized for $x = 2^{-\alpha-2}$ and hence, when applying (13.1), we get

$$\begin{aligned} P_{clear} &\geq 2^{-\alpha-2} (1/4)^{2^{-\alpha-2} + \frac{1}{2^{\alpha-1}}} \cdot (1/4)^{\frac{3\mu}{2^\alpha}} \\ &= 2^{-\alpha-2} (1/4)^{\frac{\frac{9}{4} + 3\mu}{2^\alpha}}. \end{aligned}$$

The probability P_x that *none* of the $\lambda \log n$ time slots $t \in \mathcal{I}$ is a clearance is therefore at most $P_x \geq (1 - P_{clear})^{\lambda \log n} \in O(n^{-3})$ by the definitions of λ. Notice that the reason for defining $\alpha = 6.4$ is that this value maximizes P_{clear}.

Unfortunately, the argument that in every critical interval $\mathcal{I}$ a clearance occurs with probability $1 - O(n^{-3})$ is not sufficient. Potentially, the number

of intervals $\mathcal{I}$ could be infinitely large, rendering the high probability result useless. However, the probability that in the first μ intervals $\mathcal{I}$ in D_i, there is at least one *without a clearance* is at most $O(n^{-3})$. By Lemma 13.5, there are no more than μ clearances in D_i with probability $1 - O(n^{-2})$ as long as all properties hold. Thus, there is no time slot t_1 in D_i during the execution of Algorithm 13.1 with probability at least $1 - O(n^{-2})$. Because the same argument can be applied for all D_i and because all $v \in V$ are covered by at most n disks, the claim holds for all disks with probability $1 - O(n^{-1})$. □

We continue by showing that Property 13.2 holds under the assumption that the two other Properties hold.

Theorem 13.7. *Assume Property 13.2 is among the first properties to be violated and let t_2 be the first time slot in which the violation occurs. The probability that there exists such a time slot t_2 during the execution of Algorithm 13.1 is at most $P_{fail} \in O(n^{-1})$.*

Before proving Theorem 13.7, we introduce some notation and establish a key lemma. Assume that T_c is an interval either between a) two subsequent independent clearances in a disk D_i, or b) between a clearance and the end of the algorithm, or c) between a clearance and time slot t_2 (i.e., the first violation of Property 13.2), depending on which comes first. Further, let t_c be the clearance that has initiated T_c. We show that the probability of Property 13.2 being violated in this interval (i.e., $t_2 \in T_c$) is $1 - O(n^{-3})$. By Lemma 13.1, there is no new candidate emerging in D_i in the interval $[t_c, \ldots, t_c + 4\mu\delta \log^2 n]$. We therefore need to analyze only the interval $[t_c + 4\mu\delta \log^2 n, \ldots, t_q]$, where t_q is the time slot of a) the subsequent clearance, b) the end of the algorithm, or c) time slot t_2.

Let a *failure* be a time slot in which at least one new candidate in D_i emerges, but no clearance occurs. The next lemma bounds the number of failures.

Lemma 13.8. *There are no more than $\frac{1}{6e\tau} \log n$ failures in D_i in the interval $[t_c + 4\mu\delta \log^2 n, \ldots, t_q]$ with probability $1 - n^{-3}$.*

Proof. We prove that before $\frac{1}{6e\tau} \log n$ failures can occur, there is at least one clearance with high probability. The argument is completed by the fact that, by definition, t_q must take place before or at the time of such a clearance.

We define the following events. $\mathcal{E}_c(t)$ denotes the event of a clearance in D_i at time slot t and $\mathcal{E}_0(t)$ is the event of no node in $\mathcal{A}_i$ transmitting in time slot t. Observe that $\mathcal{E}_c(t)$ can only be true if $\mathcal{E}_0(t)$ is false. In the sequel, we want to find a bound on the probability $P[\mathcal{E}_c(t)|\overline{\mathcal{E}_0}(t)]$. Clearly, if in a time slot exactly one node in D_i sends and no other node in $E_i^{1.5}$ sends, then a clearance occurs. Hence, $P[\mathcal{E}_c(t)|\overline{\mathcal{E}_0}(t)] \geq P[\mathcal{E}_1(t)|\overline{\mathcal{E}_0}(t)] \cdot P[\mathcal{E}_e(t)]$ where $\mathcal{E}_1(t)$ is the event of *at most one* node sending in $\mathcal{A}_i$, and $\mathcal{E}_e(t)$ is the event of no node sending in $E_i^{1.5} \setminus \mathcal{A}_i$. It will be convenient to state the above expression

in terms of $\mathcal{E}_+(t)$ which is the event that 2 or more nodes in $\mathcal{A}_i$ send. Thus,

$$\begin{aligned} P[\mathcal{E}_c(t)|\overline{\mathcal{E}_0}(t)] &\geq P[\mathcal{E}_e(t)] \cdot (1 - P[\overline{\mathcal{E}_1}(t)|\overline{\mathcal{E}_0}(t)]) \\ &= P[\mathcal{E}_e(t)] \cdot (1 - P[\mathcal{E}_+(t)|\overline{\mathcal{E}_0}(t)]) \\ &= P[\mathcal{E}_e(t)] \cdot \left(1 - \frac{P[\mathcal{E}_+(t)]}{P[\overline{\mathcal{E}_0}(t)]}\right), \end{aligned}$$

because of $P[\overline{\mathcal{E}_0}(t)|\mathcal{E}_+(t)] = 1$. By the definition of t_q (which is t_2 or earlier), we can assume that in the interval $[t_c + 4\mu\delta \log^2 n, \ldots, t_q]$, all three properties hold. Thus, we are allowed to reuse some results that we have established based on the assumption that all three properties hold. First, we need a bound on $P[\mathcal{E}_e(t)]$ from the proof of Theorem 13.6:

$$P[\mathcal{E}_e(t)] \geq \prod_{v \in E_i^{1.5}} (1 - p_v) \geq (1/4)^{\frac{3\mu}{2^\alpha}}.$$

For succinctness, let $\mathcal{X}_A = \sum_{v \in \mathcal{A}_i} p_v$. We obtain the following lower bound for $P[\overline{\mathcal{E}_0}(t)]$,

$$\begin{aligned} P[\overline{\mathcal{E}_0}(t)] &= 1 - \prod_{v \in \mathcal{A}_i} (1 - p_v(t)) \\ &\geq 1 - (1/e)^{\sum_{v \in \mathcal{A}_i} p_v} \geq 1 - (1/e)^{\mathcal{X}_A}. \end{aligned}$$

Finally, we consider $P[\mathcal{E}_+(t)]$,

$$\begin{aligned} P[\mathcal{E}_+(t)] &\leq P[\overline{\mathcal{E}_0}(t)] - \sum_{v \in \mathcal{A}_i} \left(p_v \prod_{w \in \mathcal{A}_i \setminus \{v\}} (1 - p_w) \right) \\ &\leq 1 - \prod_{v \in \mathcal{A}_i} (1 - p_v) - \sum_{v \in \mathcal{A}_i} p_v \cdot (1/4)^{\sum_{v \in \mathcal{A}_i} p_v} \\ &\leq 1 - (1/4)^{\sum_{v \in \mathcal{A}_i} p_v} - \sum_{v \in \mathcal{A}_i} p_v \cdot (1/4)^{\sum_{v \in \mathcal{A}_i} p_v} \\ &\leq 1 - (1 + \mathcal{X}_A)\,(1/4)^{\mathcal{X}_A}. \end{aligned}$$

Plugging everything together, the probability $P[\mathcal{E}_c(t)|\overline{\mathcal{E}_0}(t)]$ that there is a clearance if a new candidate emerges in D_i is at least $P[\mathcal{E}_c(t)|\overline{\mathcal{E}_0}(t)] \geq \mathcal{Q}$ where $\mathcal{Q}$ is

$$\mathcal{Q} = (1/4)^{\frac{3\mu}{2^\alpha}} \cdot \left(1 - \frac{1 - (1 + \mathcal{X}_A)\,(1/4)^{\mathcal{X}_A}}{1 - (1/e)^{\mathcal{X}_A}}\right).$$

By Property 13.1, we know that the expression $\mathcal{X}_A = \sum_{v \in \mathcal{A}_i} p_v$ is in the range $[0, \ldots, 2^{-\alpha}]$. Under this condition, the above function is minimized

for $\mathcal{X} = 2^{-\alpha}$, hence $P[\mathcal{E}_c(t)|\overline{\mathcal{E}_0}(t)] \geq 0.23 \cdot (1/4)^{\frac{3\mu}{2^\alpha}}$. Finally, if $t_2 \in T_c$, the probability P_f that there are more than $\frac{1}{6e\tau} \log n$ failures in D_i in the interval $[t_c + 4\mu\delta \log^2 n, \ldots, t_q]$ is asymptotically in

$$P_f \quad \leq \quad (1 - P[\mathcal{E}_c(t)|\overline{\mathcal{E}_0}(t)])^{\frac{1}{6e\tau} \log n} \quad \leq \quad O(n^{-3})$$

by the definition of τ.

That is, with probability $1 - O(n^{-3})$, if as many as $\frac{1}{6e\tau} \log n$ failures had occurred before t_q, there would have been another clearance *before* t_q, contradicting the definition of t_q. □

Proof of Theorem 13.7. We begin by showing that in expectation, there are only a constant number of new candidates emerging in D_i per failure time slot. We denote by $C(t)$ the number of active nodes that send at time t (new candidates) and write $\mathcal{E}_f(t)$ for the event of a failure. The conditional expected value of $C(t)$ given a failure is $E[C(t)|\mathcal{E}_f(t)] \leq E[C(t)] + 2 \leq \sum_{v \in \mathcal{A}_i} p_v + 2$. Because we can assume Property 13.1 to hold, this is at most $E[C(t)|\mathcal{E}_f(t)] \leq 2^{-\alpha} + 2$.

In the following, we bound the number of candidates $C(T_C)$ emerging during T_C in the case in which during the interval $T_C = [t_c + 4\mu\delta \log^2 n, \ldots, t_q]$, there are no more than $\frac{1}{6e\tau} \log n$ failures. Observe that bounding $C(T_C)$ suffices to prove the theorem because by Lemma 13.4, all candidates existing at time t_c are covered by the time $t_c + 4\mu\delta \log^2 n$ with probability $1 - n^{-2}$. Consequently, we only need to bound the number of *new* emerging candidates when analyzing the interval $[t_c + 4\mu\delta \log^2 n, \ldots, t_2]$.

The following random experiment allows to derive a high probability bound on $C(T_C)$. Consider random variables X_{ij} for $i = 1 \ldots n$ and $j = 1 \ldots |C|$, where $|C|$ is defined as the number of failures in T_C, formally $|C| = |\{t \in T_C | \mathcal{E}_f(t)\}|$. Further, we define $X := \sum_{j=1}^{|C|} \sum_{i \in \mathcal{A}_i(t_j)} X_{ij}$ as the sum of all X_{ij}. The semantic meaning of X_{ij} is that $X_{ij} = 1$, if node i transmits (and becomes a candidate) in the j^{th} failure of T_C, and $X_{ij} = 0$ otherwise. Therefore, X represents an upper bound on the number of new candidates emerging in D_i during T_C. Considering the X_{ij} as being independently distributed Bernoulli trials is not precise because of dependencies between different X_{ij}. Specifically, $X_{ij} = 1 \Rightarrow X_{ij'} = 0$, for all $j' > j$ because when transmitting, an active node becomes a candidate. Note that these dependencies cause X to be strictly smaller or equal as compared to the case in which all X_{ij} that are depending on previous events were chosen randomly and independently with an arbitrary probability distribution. Thus, when assuming all X_{ij} to be independent Bernoulli trials, X is an upper bound for $C(T_C)$.

We know from the above argument, that in expectation, at most $2^{-\alpha} + 2$ active nodes transmit per failure. With our assumption that there are no

more than $\frac{1}{6e\tau}\log n$ failures, we get

$$\begin{aligned} E[X] &= \sum_{j=1}^{|C|} E\left[\sum_{i\in\mathcal{A}_i(t_j)} X_{ij}\right] = \sum_{j=1}^{|C|} E[C(t_j)|\mathcal{E}_f(t_j)] \\ &\leq (2^{-\alpha}+2)\frac{1}{6e\tau}\log n < \frac{1}{(2e+1)\tau}\log n. \end{aligned}$$

As mentioned before, assuming X_{ij} to be randomly and independently distributed Bernoulli variables yields an upper bound on $C(T_C)$, i.e., $E[C(T_C)] \leq E[X]$. Hence, we can use the Chernoff bound with $E[X] = \frac{\log n}{(2e+1)\tau}$. In particular, the probability P_x that X is larger than $(2e+1)\frac{\log n}{(2e+1)\tau} = \tau^{-1}\log n$ is at most

$$P[C(T_C) > \tau^{-1}\log n] \leq \left(\frac{e^{2e}}{(2e+1)^{2e+1}}\right)^{\frac{\log n}{(2e+1)\tau}} \in O(n^{-2}).$$

That is, *if* there are at most $\frac{1}{6e\tau}\log n$ failures in T_C, then at most $\tau^{-1}\log n$ candidates emerge in T_C with probability $1-O(n^{-2})$. As shown, this bound suffices to prove that Property 13.2 is not violated in T_C with probability $1-O(n^{-2})$. On the other hand, we know by Lemma 13.8 that the probability of having *more than* $\frac{1}{6e\tau}\log n$ failures in T_C is $1-O(n^{-2})$. That is, the probability that in an arbitrary interval T_C after a clearance, Property 13.2 is indeed violated *before* the next clearance is at most $1-2\cdot O(n^{-2}) = 1-O(n^{-2})$.

Now, consider the first μ intervals T_C in *every disk* D_i. The probability that there is at least one interval in which there are more than $\tau^{-1}\log n$ new candidates is at most $n\mu\cdot O(n^{-2}) \in O(n^{-1})$. That is, with probability $1-O(n^{-1})$, Property 13.2 is *not violated* after the first μ intervals T_C in every disk D_i. By Lemma 13.5, there are no more than μ clearances (and hence inter-clearance intervals T_C) per disk with probability $1-O(n^{-2})$ if all three Properties hold, which is the case before t_2. Thus, Property 13.2 is *not* among the first Properties to be violated with probability $1-O(n^{-1})$. □

Finally, we prove the correctness of Property 13.3.

Theorem 13.9. *Assume Property 13.3 is among the first properties to be violated and let t_3 be the first time slot in which the violation occurs. The probability that there exists a time slot t_3 during the execution of Algorithm 13.1 is at most $P_{fail} \in O(n^{-2})$.*

Proof. We show that if a node joins $\mathcal{L}$, the counter values c_v of all neighboring candidates are at least $\beta\log n$ away from the threshold that enables to join $\mathcal{L}$. Applying Lemma 13.2 then concludes the proof.

Let v_v be the node that violates Property 13.3 at time $t_v = t_3$ and let v_m be the neighbor of v_v that has previously joined $\mathcal{L}$, say at time $t_m \leq t_v$. We

claim that at time t_m, the counter value c_x of all neighbors $v_x \in \Gamma(v_m)$ of v_m (including v_v) is at most $\delta \log^2 n - \beta \log n$.

By the definition of the algorithm, v_m must have started increasing its counter by the time $t_m - \delta \log^2 n$. Similarly, every potential node v_x that ends up having a counter larger than $\delta \log^2 n - \beta \log n$ at time t_m must have started increasing its c_x by the time $t_m - \delta \log^2 n + \beta \log n$. By the definition of the critical range $\beta \log n$ in Receive Trigger 2, such a node v_x has not received a message $M_{\mathcal{C}}$ from v_m in the interval $[t_m - \delta \log^2 n + \beta \log n, \ldots, t_m]$ because if it had, it would have reset its counter c_x.

The probability P_t that $v_x \in D_i$ receives a message $M_{\mathcal{C}}$ from v_m in an arbitrary time slot t in the interval $[t_m - \delta \log^2 n + \beta \log n, \ldots, t_m]$ is at least

$$\begin{aligned} P_t &\geq p_m \cdot \prod_{v \in E_i^{1.5}} (1 - p_v) \\ &\geq q_C \prod_{D_j \in E_i^{1.5}} (1/4)^{\sum_{v \in \mathcal{C}_j \cup \mathcal{A}_j} p_v + \sum_{v \in \mathcal{L}_j} p_v} \\ &\underset{\text{Properties 13.1, 13.2}}{\geq} \frac{\tau}{2^\alpha \log n} \left[(1/4)^{\frac{1}{2^{\alpha-1}} + \sum_{v \in \mathcal{L}_j} p_v} \right]^\mu . \end{aligned}$$

Because v_v is the first node violating Property 13.3, it holds that before t_m, the set $\mathcal{L}$ forms a correct independent set. Therefore, due to Fact 13.1, $\sum_{v \in \mathcal{L}_j} p_v \leq \frac{1}{2^\alpha}$, and hence $P_t \geq 2^{-\alpha} \tau (\log n)^{-1} (1/4)^{\frac{3\mu}{2^\alpha}}$. The probability P_n that a node v_x does *not* receive any message $M_{\mathcal{C}}$ from v_m during the $\delta \log^2 n - \beta \log n$ remaining time slots before t_m joins $\mathcal{L}$ is

$$\begin{aligned} P_n &\leq \left(1 - \frac{\tau}{2^\alpha \log n} (1/4)^{\frac{3\mu}{2^\alpha}} \right)^{\delta \log^2 n - \beta \log n} \\ &\leq \left(1 - \frac{\frac{\tau \log n}{2^\alpha}}{\log^2 n} (1/4)^{\frac{3\mu}{2^\alpha}} \right)^{\frac{1}{2} \delta \log^2 n} \\ &\leq e^{-\frac{1}{2} \beta \log n \cdot \frac{\tau}{2^\alpha} (1/4)^{\frac{3\mu}{2^\alpha}}} \in O(n^{-4}). \end{aligned}$$

Because there are n^2 pairs of nodes $(v_m, v_x) \in V \times V$, the probability that the "counter-difference" claim holds for all nodes v_m and v_x is at least $1 - O(n^{-2})$.

We now have all ingredients to prove the theorem. Assume for contradiction that v_v is the first node to violate the MIS condition (Property 13.3) at time t_3 and let v_m be v_v's neighbor that has correctly joined $\mathcal{L}$ at time $t_m \leq t_3$. By definition of v_v, Property 13.3 as well as the two other properties holds until $t_3 - 1$. Therefore, we can apply the result obtained above. In particular, with probability $1 - O(n^{-2})$, there are at least $\beta \log n$ time slots between v_m and any potential node v_v causing the violation. Because all properties hold before t_3, it follows from Lemma 13.2 that v_v receives a message $M_{\mathcal{C}}$ by $v_m \in \mathcal{L}$ with probability $1 - O(n^{-3})$. Hence, the probability that there exists a time slot t_3 is bounded by $O(n^{-2})$. □

When the first node wakes up, all three Properties are valid. Theorems 13.6, 13.7, and 13.9 show that none of them is the first to be violated, thus establishing the algorithm's correctness. For the running-time, we show that *every node* decides in time $O(\log^2 n)$ whether to become an MIS node or a slave.

Theorem 13.10. *In a unit disk graph G, Properties 13.1, 13.2, and 13.3 hold with probability $1 - O(n^{-1})$. Particularly, the set $\mathcal{L}$ as computed by Algorithm 13.1 is a correct maximal independent set with probability $1 - O(n^{-1})$.*

Proof. By Theorem 13.6, Property 13.1 is not the first to be violated with probability $1-O(n^{-1})$. Similarly, by Theorems 13.7 and 13.9, Properties 13.2 and 13.3 are not the first to be violated with probabilities $1-O(n^{-1})$ and $1-O(n^{-2})$, respectively. Hence, with probability $(1-O(n^{-1}))^2 \cdot (1-O(n^{-2})) \in 1 - O(n^{-1})$, none of the three properties is violated during the execution of the algorithm. If all three Properties hold, Property 13.3 implies that with probability $1 - O(n^{-1})$, the resulting set $\mathcal{L}$ forms a correct independent set. The maximality of the independent set stems from the fact that a node joins set $\mathcal{S}$ only upon receiving a message $M_{\mathcal{L}}$ from a neighboring MIS node. That is, every non-MIS node has at least one MIS node in its neighborhood. □

Theorem 13.11. *In a unit disk graph G, every node $v \in V$ decides irrevocably whether it joins set $\mathcal{L}$ or $\mathcal{S}$ within time $O(\log^2 n)$ after its wake-up with probability $1 - O(n^{-1})$.*

Proof. Consider an arbitrary node $v \in D_i$ and let $T_{\mathcal{W}}$, $T_{\mathcal{A}}$, and $T_{\mathcal{C}}$ be the total time node v spends in the corresponding state during its execution of Algorithm 13.1. Assume that all three Properties hold. Once node v becomes a candidate at time t_v, it will decide to become an MIS node or a slave within time $t_v + 4\mu\delta \log^2 n$ by Lemma 13.4 with probability $1-O(n^{-2})$. Hence, $T_{\mathcal{C}} \leq 4\mu\delta \log^2 n$. It thus remains to bound the time that v spends in states $\mathcal{W}$ and $\mathcal{A}$.

If v does not receive a message $M_{\mathcal{A}}$ from a neighboring node for $4\mu\delta \log^2 n$ time slots after its wake-up (or after being reset to state $\mathcal{W}$ in Receive Trigger 1), it becomes active and joins $\mathcal{A}$. Unless it receives a message $M_{\mathcal{A}}$ thereafter, its sending probability reaches the value $p_v(t) = 2^{-\alpha-2}$ at most $(\log n - 1) \cdot \lambda \log n$ time slots after becoming active. This is because the sending probability is initially $2^{-\alpha-1}/n$ and is doubled once per $\lambda \log n$ time slots. So, either there is a node $w \in \Gamma(v)$ whose message $M_{\mathcal{A}}$ v has received and who subsequently becomes a candidate, or the sending probability of $v \in \mathcal{A}$ exceeds the value $p_v(t) = 2^{-\alpha-2}$. If the latter is the case, conditions (13.1) and (13.2) are fulfilled. It follows by the same argument as in the proof of Theorem 13.6 that there is a clearance in the subsequent $\lambda \log n$ time slots with probability $1 - O(n^{-3})$.

Putting things together, it holds that $4\mu\delta \log^2 n + \lambda \log^2 n$ time slots after wake-up or a reset because of Receive Trigger 1, there exists a node $w \in \Gamma(v)$ that becomes a candidate, say at time t_c. If $w = v$, we are done because v joins $\mathcal{C}$ and $T_{\mathcal{C}} \leq 4\mu\delta \log^2 n$. If $w \neq v$, a new MIS node appears in the

interval $[t_c - \beta \log n, \ldots, t_c + \delta \log^2 n]$ in $E_i^{3.5}$ with probability $1 - O(n^{-2})$ by Lemma 13.3. Because there can be at most 4μ MIS nodes in $E_i^{3.5}$, the total time spent by v in states $\mathcal{W}$ and $\mathcal{A}$ is bounded by

$$T_{\mathcal{W}} + T_{\mathcal{A}} \;\leq\; 4\mu \cdot (4\mu\delta + \lambda) \log^2 n \;\in\; O(\log^2 n)$$

Together with the above bound on $T_{\mathcal{C}}$, this shows that *if* all three Properties hold, every node v decides within time $O(\log^2 n)$ upon its wake-up with probability $(1 - O(n^{-2}))^n \in 1 - O(n^{-1})$. By Theorem 13.10, all three properties hold with probability $1 - O(n^{-1})$ which concludes the proof. □

Theorems 13.10 and 13.11 show that with high probability, Algorithm 13.1 computes a correct maximal independent set in time $O(\log^2 n)$.

Chapter 14

Application: Deployment of Sensor Networks

In this Chapter, we show how the primitives introduced in Chapters 12 and 13 can be applied to support energy-efficient deployment of wireless sensor networks. One of the key characteristics of *sensor networks* is that individual sensor nodes have a limited, typically non-renewable power supply and, once deployed, must work unattended. In view of the scarcity of energy, an economical and frugal management of this resource is essential for prolonging network lifetime and availability.

The search for energy-efficient solutions has lead to numerous algorithms and protocols that strike for the goal of reducing the energy-consumption of an operational sensor network. For instance, there have been various proposals for energy-efficient medium access control (MAC) protocols [201, 214, 237, 240], routing algorithms [27, 42], topology control and clustering [67, 122, 164, 231], broadcasting [176, 228], or data gathering/dissemination [106, 225, 242]. The *non-operational phase* (i.e., the actual deployment) of sensor networks, however, has not been studied with the same zeal, even though in many applications, a crucial loss of energy occurs already *before* the sensor network reaches its operational state.

Consider a water (or power, gas, etc.) metering network for an apartment complex. Each apartment is equipped with a water metering sensor. At midnight, all sensors wake up for a few seconds, the water consumption of each apartment is sent to a base station in multi-hop fashion, and all sensors go back to sleep for another 24 hours. In the operational phase such a sensor network features a gargantuan sleep/awake ratio, allowing even conventional batteries to last several years. In order to reach such a long lifetime, the node's duty cycle must be significantly below 1%. However, the deployment of the sensor nodes might take days or weeks. With a naive deployment protocol, say, when nodes stay awake until the entire system is deployed, the battery of the node deployed first might be drained before

the network even becomes operational. This highlights a problem that is particularly pronounced in settings in which the node's duty cycle during the operational phase is small, but the deployment takes long. Many other applications featuring such a time-consuming deployment phase exist, e.g. vehicle tracking, or monitoring large-scale industrial processes.

Generally, once all sensor nodes are fully deployed, the network should make the transition from the deployment phase to the operational phase as quickly as possible. In particular, we might like to externally trigger a *network discovery* procedure that allows verifying the operability of the newly deployed network (e.g. detect faulty sensor nodes). Clearly, simple solutions to invoke such a system initialization would be to manually switch on all nodes once the deployment phase is completed, or to set a timer when physically deploying each node. Unfortunately, in many practical settings, neither of these hands-on solutions is practicable. First, nodes may be deployed in remote or hostile environment in which switching on nodes manually after all nodes are deployed may be impossible. Moreover, in application scenarios featuring a time-consuming deployment phase, predicting the exact duration of the deployment process is usually hard, hence ruling out the possibility of employing a solution based on predefined timers.

So how can the information about the beginning of the operational phase be distributed among the network nodes? Typically, this information is supposed to be *broadcasted by the nodes* in a multi-hop way through the entire network such that, eventually, every sensor node will know that the system is now ready to start its operational phase. Specifically, one or several nodes (in typical sensor network applications, this is usually the base station) are triggered externally. These nodes then try to inform their neighbors, who in turn inform their neighbors, and so forth. We call this externally triggered event that sets off the information broadcast the *launching point*.

Ideally, each node should remain in some kind of energy-saving *sleep mode* for the entire duration of the deployment phase preceding the launching point. In sleep mode, nodes do neither send data packets nor listen for incoming messages [215]. The problem is however that individual nodes do not know the exact time of the launching point, or the duration of the deployment phase. As a consequence, in order to learn about the arrival of the launching point from neighbors, a node must periodically leave the sleep mode and listen for incoming messages.[1] This observation establishes a trade-off between the energy consumption of nodes during the deployment phase and the rapidity of the transition to the operational phase after the launching point. Neither of the two extremes, *always asleep* and *always awake* during the deployment, is satisfying; any decent protocol is in-between.

The delay vs. energy trade-off is important in sensor networks beyond the deployment phase, especially in sensor networks that concentrate on discovering rare events, e.g. sensor networks for seismic surveillance in earthquake

[1] Obviously, the problem could be elegantly solved using very low power "trigger" circuits, which operate continuously on small power budgets, and wake up more power-hungry circuits only upon receipt of a suitable signal from a neighboring node. Unfortunately, currently available standard hardware such as the Mica2 [123] wireless sensor nodes do not offer this functionality, and we therefore do not consider this option.

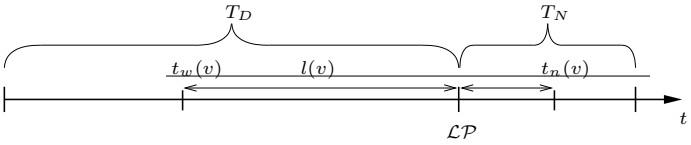

Figure 14.1: The deployment phase is of length T_D, the notification phase is of length T_N.

and rubble zones. The pronounced "event" character of such rare events leads to exactly the deployment-problem trade-off. Namely, since events occur rarely, sensor nodes should be in sleep mode as often as possible to save energy. These energy savings, however, come at the cost of a prolonged reaction time once a rare event occurs.

In Section 14.1, we model the deployment problem in a way that allows to compare different protocols, independent of application specific parameters such as node distribution or deployment pattern. After evaluating the performance of two simple entirely unstructured approaches in Section 14.2, we present an improved "semi-structured" protocol, which is based on the MIS algorithm of Chapter 13.

The communication model used for analyzing the various protocols is again the unstructured radio network model introduced in Chapter 11. In particular, nodes may wake up at any time without knowledge about their neighborhood and without access to a global clock. Further, we assume that all nodes are deployed at the time of the launching point, i.e., before the transition to the operational phase.

14.1 The Deployment Problem

We divide the *non-operational phase* of a sensor network into two parts, the *deployment phase* and *notification phase* as shown in Figure 14.1. In the deployment phase, sensor nodes are physically positioned at their intended locations. Once this is done for all sensor nodes, the notification phase is triggered, in which the aim is to inform all nodes about the system being up and running. The transition to this second phase is induced by an externally triggered event—at the *launching point* $\mathcal{LP}$—that is received by at least one node in the network. During the notification phase, we call a node *notified* if it has already received the notification message, and *unaware* otherwise. At the launching point, at least 1 node is notified whereas at most $n-1$ nodes are unaware.

During the deployment phase, an algorithm may build an initial structure which can help speeding up the notification process later on. On the other hand, the building and maintenance (incorporating newly awakening nodes into a tree, for example) of such a structure requires the nodes to stay awake longer and thus spend more energy. In order to enable a fair comparison

between different algorithmic approaches, the problem definition has to be general enough to account for these various possibilities.

The total energy consumption of a node v in a deployment algorithm $\mathcal{A}$ can be divided into two parts, the *initialization energy* and the *maintenance energy*. The initialization energy $e_{init}(v)$ is the total amount of energy used by v to initially join a desired structure (e.g., decide whether it is a clusterhead or become a part of a tree). A node's initialization energy accrues only once, regardless of the length of the deployment phase. In contrast, the maintenance energy $e_m(v)$ denotes the total amount of energy used by v once it has been properly initialized. Specifically, the maintenance energy $e_m(v)$ encompasses the node's periodic wake-up necessary to learn about the launching point. If $e_m(v)$ is small, the node will require a long time before learning about the $\mathcal{LP}$, thus slowing down the notification phase. Depending on the nature of the algorithm, $e_m(v)$ may comprise additional aspects. Consider, for instance, a protocol that is based on maintaining a tree-structure which allows for rapid event dissemination during the notification phase. In this case, already initialized nodes periodically send messages in order to inform neighbors that may have woken up in the meantime, thus enabling their integration into the tree.

Formally, let T_D and T_N be the length of the deployment phase and notification phase, respectively. Further, $t_w(v)$ denotes the wake-up point of node v. The time v is active before the launching point is $\ell(v) = t(\mathcal{LP}) - t_w(v)$. Since we consider asynchronous wake-up with an imaginary adversary determining each node's wake-up point (and hence $\ell(v)$), we must consider the *average maintenance energy* $a_m(v) = e_m(v)/\ell(v)$. This value describes the maintenance energy used by a node v for a single time slot between its wake-up and the LP. Note that $a_m(v)$ is independent of a node's $t_w(v)$, $\ell(v)$, or the time of the launching point, because $a_m(v)$ considers only periodical maintenance costs, i.e., no initialization costs.

We still have to come up with a measure for the algorithm's *energy efficiency* that takes into account both the maintenance and the initialization costs, but remains independent of the specific wake-up pattern. For that, we define the energy efficiency of an algorithm $\mathcal{A}$ with regard to a deployment phase of length T_D, denoted by $E(\mathcal{A}, T_D)$, as the *average energy of algorithm $\mathcal{A}$ per node and per time slot*. That is, an algorithm in which all nodes listen in every time slot has energy efficiency equal to 1, whereas the algorithm that lets all nodes sleep all the time has energy efficiency 0. With this definition, the measure of an algorithm's energy efficiency does not depend on the particular wake-up pattern of a given problem instance, capturing instead the characteristic of the algorithm itself.

The two main quality measures of a deployment algorithm $\mathcal{A}$ are formally defined as follows.

Definition 14.1. *Let $\mathcal{A}$ be a deployment algorithm and let T_D be the length of the deployment phase before the launching point. Also, let $f(n)$ be a minimal function such that with probability at least $1 - 1/n$, it holds that $T_N \leq f(n)$. The algorithm's energy and time efficiency, $E(\mathcal{A}, T_D)$ and $T(\mathcal{A}, T_D)$, are*

defined as

$$\begin{aligned} E(\mathcal{A}, T_D) &:= \frac{1}{n \cdot T_D} \sum_{v \in V} (e_{init}(v) + T_D \cdot a_m(v)), \\ T(\mathcal{A}, T_D) &:= f(n). \end{aligned}$$

Note that the definition of $E(\mathcal{A}, T_D)$ corresponds to the intuitive notion of *energy efficiency* given above. Particularly, the terms $T_D \cdot a_m(v)$ and $e_{init}(v)$ describe a node v's maintenance and initialization energy during a deployment phase of length T_D, respectively. Adding up these values over all nodes and dividing by $\frac{1}{n \cdot T_D}$, the number of nodes and time slots, leads to the energy efficiency $E(\mathcal{A}, T_D)$. As for the second measure, an algorithm has time efficiency $f(n)$ (for instance n^2) if with high probability, all nodes are notified $f(n)$ time slots after the launching point.

Definition 14.1 allows us to compare deployment algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$ in two ways. First, we can fix the notification time $f(n)$ and compare the algorithm's energy requirements. That is, we demand two algorithms to finish the notification period within the same amount of time and then compare which algorithm requires more energy during the deployment phase in order to ensure that all nodes are notified within $f(n)$, i.e., $T_N \leq f(n)$. Alternatively, we can fix the energy consumption $E(\mathcal{A}_1, T_D)$ and $E(\mathcal{A}_2, T_D)$, respectively, of both algorithms and then compare the resulting length of the notification phase. Clearly, both comparison methodologies are two sides of the same coin; they both describe the inherent trade-off between energy efficiency and the rapidity of information dissemination.

14.2 Simple Algorithms

We begin by analyzing the trade-offs of two simple algorithms in our framework. The first algorithm is the so-called *birthday algorithm* proposed in [174] which, among other applications, can be used as an algorithm for the deployment of sensor networks. Originally, $\mathcal{A}_{birth}$ has been designed and analyzed for neighborhood discovery, i.e., not for the deployment problem as considered in this paper. The second algorithm enhances the birthday algorithm with a standard trick in order to achieve significantly better trade-offs.

Birthday Algorithm

The birthday algorithm $\mathcal{A}_{birth}$ proposed in [174] is conceptually simple: During the deployment phase, before being notified, a node v listens in each time slot with probability p_L and sleeps with probability $1 - p_L$. Once v has learned about the launching point, it transmits with probability $p_T := 1/n$ and listens with probability p_L.

What is the trade-off exhibited by $\mathcal{A}_{birth}$ with regard to the definitions given in Section 14.1? Let $f(n)$ be a function such that $T_N(\mathcal{A}_{birth}) \leq f(n)$

with high probability. Given this constraint, we want to optimize the algorithm's energy efficiency. The achievable trade-off is expressed in the following theorem.

Theorem 14.1. *Let $f(n)$ be a function such that the* birthday algorithm *$\mathcal{A}_{birth}$ has time efficiency $T(\mathcal{A}_{birth}, T_D) \leq f(n)$. For general network graphs and for arbitrary T_D, $\mathcal{A}_{birth}$'s energy efficiency is*

$$E(\mathcal{A}_{birth}, T_D) \in \Theta\left(\frac{n^2}{f(n)}\right).$$

Proof. The birthday algorithm does not require any initialization and therefore, $e_{init}(v) = 0$, for all $v \in V$. The average maintenance energy for each node corresponds directly to the listening probability, i.e., $a_m(v) = p_L$. Hence, the algorithm's energy efficiency is

$$E(\mathcal{A}_{birth}, T_D) = \frac{1}{n \cdot T_D} \sum_{v \in V} (T_D \cdot p_L) = p_L.$$

Consider the network graph $G_b = (V_b, E_b)$ consisting of nodes $v_1, \ldots, v_n$ positioned in a line, i.e., v_i is a neighbor of v_j iff $j = i + 1$ and $1 < j \leq n$. Recall that in the unstructured radio network model, nodes have no knowledge about the topology of the network. Finally, let v_0 be the node which is externally triggered at the launching point. By the construction of G_b, the information about the arrival of the launching point has to traverse the entire network in a hop-by-hop fashion. We call a time slot t *successful*, if there is a notified node v_i that transmits in t and its unaware neighboring node v_{i+1} listens at the same time. In order to pass the notification through the entire chain, a minimum of $n - 1$ successful time slots are required, and the probability P_{suc} that a time slot t is successful is $P_{suc} = p_L \cdot p_T$.

In total, the algorithm is allowed to use $f(n)$ time slots and the broadcast has to succeed with probability at least $1 - 1/n$. Given these constraints, we want to minimize p_L thus optimizing $E(\mathcal{A}_{birth}, T_D)$. In expectation, the number of successful rounds is $p_L p_T f(n)$. Since we want at least $n - 1$ successes, it follows that

$$\frac{p_L f(n)}{n} = n - 1 \quad \Rightarrow \quad p_L \in \Omega\left(\frac{n^2}{f(n)}\right).$$

Finally, we show that for a large enough constant c, $p_L = cn^2/f(n)$ is enough to obtain the high probability argument. Let X be the number of successful rounds. The expected value of X is $\mu = p_L f(n)/n$. We bound the probability of having less than $n-1$ successful rounds using Chernoff Bounds as

$$\begin{aligned} P[X < n-1] &= P\left[X < \left(1 - \left(1 - \frac{n(n-1)}{p_L f(n)}\right)\right) \frac{p_L f(n)}{n}\right] \\ &< e^{-\frac{p_L f(n)}{2n}\left(1 - \frac{n(n-1)}{p_L f(n)}\right)^2} = e^{-\frac{cn}{2}\left(1 - \frac{1}{c}\right)^2}, \end{aligned}$$

Algorithm $\mathcal{A}_{uni}$

upon wake-up do:
1: listen with probability p_L, otherwise sleep
upon notification do:
2: **for** $i := \lceil \log n \rceil + 1$ to 1 by -1 **do**
3: $p_T := 1/2^i$
4: **for** $\frac{c(\lceil \log n \rceil + 1)}{p_L}$ time slots **do**
5: transmit message with probability p_T
6: **end for**
7: **end for**

which is smaller than $1/n$ for a suitably large constant c. Notice that setting p_L to a value strictly smaller, i.e., $p_L \in o(n^2/f(n))$ renders the exponent positive thus not yielding the desired result. □

Uniform Algorithm

The birthday algorithm does not maintain any structure and hence, features no initialization cost. Using the idea of exponentially increasing transmission probabilities already encountered in Chapter 13, the birthday algorithm's performance can be significantly improved. We call this improved version of the birthday algorithm *uniform algorithm* $\mathcal{A}_{uni}$. Algorithm $\mathcal{A}_{uni}$ has one input parameter, the listening probability p_L; c is a constant to be defined later. In comparison with the birthday algorithm $\mathcal{A}_{birth}$ analyzed in Section 14.2, $\mathcal{A}_{uni}$ exhibits a strictly better performance trade-off as stated in Theorem 14.2.

Theorem 14.2. *Let $f(n)$ be a function such that the* uniform algorithm *$\mathcal{A}_{uni}$ has time efficiency $T(\mathcal{A}_{uni}, T_D) \leq f(n)$. In any unit disk graph G and for arbitrary T_D, $\mathcal{A}_{uni}$'s energy efficiency is at most*

$$E(\mathcal{A}_{uni}, T_D) \in O\left(\frac{n \log^2 n}{f(n)}\right).$$

Proof. Like $\mathcal{A}_{birth}$, $\mathcal{A}_{uni}$ does not require any initialization and all nodes are treated uniformly. Therefore, by the same argument as in Section 14.2, we have $E(\mathcal{A}_{uni}, T_D) = p_L$. The remainder of the proof follows along the same lines as the proofs of the MIS algorithm in Chapter 13.

We define the listening probability p_L to be $p_L := cn(\lceil \log n \rceil + 1)^2/f(n)$ and seek to show that for a constant $c \geq 12$, the probability of the notification message advancing at least *one hop* in time $O(f(n)/n)$ is at least $1 - n^{-2}$. Since the diameter of the network is at most n, the theorem follows from applying the union bound.

Let $Z_{v,t}$ denote the event of node v hearing a notification message in time slot t. Consider an unaware node $v \in V$ and let t_0 be the first time slot in which at least one node in v's neighborhood $\Gamma(v)$ is notified. Starting from

this round, the sum of sending probabilities $\sum_{w\in\Gamma(v)} p_T(w)$ increases. Let t^* be the last time slot in which the sum of sending probabilities is smaller than $1/2$. Notice that it takes at most $t^* - t_0 \leq (\lceil \log n \rceil + 1) \cdot \frac{c(\lceil \log n \rceil + 1)}{p_L}$ time slots until t^* is reached.

Now, consider the time interval $\mathcal{I} = [t^* + 1, \ldots, t^* + \frac{c(\lceil \log n \rceil + 1)}{p_L}]$. During this interval, notified nodes can at most double their p_T and new nodes will transmit with the initial sending probability $p_T = \frac{1}{2n}$. At the end of this interval, the sum of sending probabilities is therefore at most

$$\sum_{w\in\Gamma(v)} p_T(w) \;\leq\; 2 \cdot \frac{1}{2} + \sum_{w\in\Gamma(v)} \frac{1}{2n} \;\leq\; \frac{3}{2}. \tag{14.1}$$

Therefore, in each time slot $t \in \mathcal{I}$, the sum of sending probabilities is at least $1/2$ and at most $3/2$. The probability $P[Z_{v,t}]$ that v receives the notification message from one of its neighbors is

$$\begin{aligned}
P[Z_{v,t}] \quad &= \quad p_L \sum_{w\in\Gamma(v)} \left(p_T(w) \cdot \prod_{\substack{q\in\Gamma(v)\\ q\neq w}} (1 - p_T(q)) \right) \\
&\geq \quad p_L \sum_{w\in\Gamma(v)} p_T(w) \cdot \prod_{q\in\Gamma(v)} (1 - p_T(q)) \\
&\underset{\text{Fact 2.1}}{\geq} \quad p_L \sum_{w\in\Gamma(v)} p_T(w) \cdot \left(\frac{1}{4}\right)^{\sum_{q\in\Gamma(v)} p_T(q)} \\
&\geq \quad \frac{3p_L}{2} \cdot \left(\frac{1}{4}\right)^{3/2} \;>\; \frac{p_L}{6}.
\end{aligned}$$

For large enough functions $f(n)$ and $p_L = cn(\lceil \log n \rceil + 1)^2/f(n)$, the probability that none of the $\frac{c(\lceil \log n \rceil + 1)}{p_L}$ time slots $t \in \mathcal{I}$ is successful is at most

$$P[\cap_{t\in\mathcal{I}} \overline{Z_{v,t}}] \quad \leq \quad \left(1 - \frac{p_L}{6}\right)^{\frac{c(\lceil \log n \rceil + 1)}{p_L}} \underset{\text{Fact 2.2}}{\leq} e^{-\frac{c}{6}(\lceil \log n \rceil + 1)} \;<\; n^{-2}.$$

Therefore, with probability exceeding $1 - n^{-2}$, the notification message is passed on at least by one hop in time

$$t^* - t_0 \;\leq\; (\lceil \log n \rceil + 1) \cdot \frac{cf(n)(\lceil \log n \rceil + 1)}{cn(\lceil \log n \rceil + 1)^2} \;=\; \frac{f(n)}{n}.$$

Consequently, the notification message reaches all n nodes within time $f(n)$ with probability at least $1 - n^{-1}$. □

The trade-off obtained by $\mathcal{A}_{uni}$ is strictly better than the one obtained by the birthday algorithm in Section 14.2. Moreover, in the case $p_L = 1$, the algorithm allows a feasible solution for functions $f(n) \in \Omega(n \log^2 n)$ as opposed to $f(n) \in \Omega(n^2)$ for the birthday algorithm.

14.3 Cluster-Based Algorithm

In this section, we show that in certain network settings, employing a *semi-structure* can render the notification of nodes during the notification phase quicker. On the other hand, installing and maintaining the structure requires additional energy during the deployment phase and hence, contrary to the algorithms in Section 14.2, the cluster algorithm $\mathcal{A}_{clu}$ has non-zero initialization costs $e_{init}(v)$. Algorithm $\mathcal{A}_{clu}$ is based on a more restricted network model. We assume that nodes are located in a doubling unit ball graph as defined in Chapter 8 (e.g., a unit disk graph). Each node is capable of adapting its transmission power in such a way that its transmission range is halved. Finally, we assume that the network's density is reasonably high, that is, there is at least one node in every disk of radius $1/4$ in the convex hull of the nodes.

The design of $\mathcal{A}_{clu}$ aims at mending the main dissipation of energy of the two previous algorithms, the lack of synchronization. If neighboring nodes had synchronized wake-up points, the notification phase would take significantly less time. Consequently, when demanding the same notification efficiency T_N, the nodes could sleep longer, thus saving energy during the deployment phase. The problem is that synchronization between neighboring nodes incurs additional set-up and maintenance costs and the question is whether these additional costs will equiponderate the gains stemming from the above mentioned notification speed-up.

Our approach is based on grouping nodes into synchronized clusters. Within a cluster, nodes wake-up at the same time. In particular, we consider an MIS Q on a graph G' in which two nodes are adjacent if their mutual distance is at most $1/2$. In other words, G' is the ball graph in which every node's transmission range is set to $1/2$. Let $s(v)$ denote the leader of node v and for $u \in Q$ let $S(u)$ refer to the set of nodes having u as their leader, i.e., $S(u) = \{v | u = s(v)\}$ for all $u \in Q$. Notice that by the lower bound on the network density dictated by the restricted model, it is guaranteed that the set Q is connected if we consider all two-hop paths in G.

When constructing the MIS on G', we employ a simple adaptation of the algorithm presented in Chapter 13. In particular, the only change to the algorithm is to increase its initial waiting time from $O(\log^2 n)$ time slots to W time slots, where the exact value of $W \in \Omega(\log^2 n)$ is to be decided later. Hence, in total, every node needs to be awake for $W + O(\log^2 n)$ time slots before deciding whether it becomes a leader (i.e., joins the MIS) or not. Subsequently, leaders transmit with a sending probability of $\Theta(\log n/W)$ (instead of $\Theta(1)$ as in Algorithm 13.1 of Chapter 13) in order to inform newly awakening nodes of their being covered. This prevents nodes that wake up later from invalidating the MIS condition.

In $\mathcal{A}_{clu}$ each leader $v \in Q$ coordinates the nodes in $S(v)$ and is responsible for their synchronized waking up. Specifically, a leader v decides on the timing of the *rendezvous windows* for its cluster; a time window during which the nodes $w \in S(v)$ are simultaneously awake. Every node $w \in S(v)$ learns the timing of these rendezvous windows from its leader v. The idea is that a notified leader can notify all nodes in its cluster at almost the same time.

Algorithm $\mathcal{A}_{clu}$: Code for non-leader u

upon wake-up do:
1: perform MIS algorithm of length $O(W + \log^2 n) \rightarrow$ decide on leader $s(u)$, receive wake-up point r_3
2: **loop**
3: sleep until next wake-up point r_3
4: for $\eta \log n$ time slots listen for notification message $\mathcal{M}_n$
5: $r_3 := r_3 + I$
6: **end loop**
upon notification do:
7: **loop**
8: upon receiving $\mathcal{M}_a(r_2)$, wait until r_2 } S_1
9: **for** $i := \lceil \log n \rceil + 1$ to 1 by -1 **do**
10: **for** $(\gamma+\eta)(\lceil \log n \rceil + 1)$ time slots **do**
11: transmit message with probability $p_T = 1/2^i$
12: upon receiving $\mathcal{M}_r$, quit for-loops } S_2
13: **end for**
14: **end for**
15: **end loop**

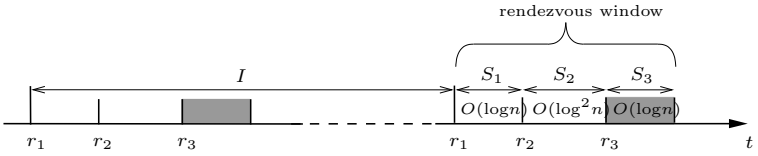

Figure 14.2: Rendezvous interval I and rendezvous window

Each rendezvous takes place in three steps as shown in Figure 14.2 and during a rendezvous, nodes transmit using their full transmission range. In the *proclamation step* S_1, leader v announces the rendezvous interval to neighboring nodes which do *not* belong to $S(v)$. The reason is that once a node is notified, it remains listening on the channel. Such a node must be able to notify neighboring leaders, even if it is in a different cluster itself (otherwise, the notification message would not broadcast through the network). In other words, the proclamation step is intended for announcing the notification across cluster boundaries. The conveyance of the notification message in the opposite direction is the aim of the second step, the *leader-notification step* S_2. In this step, already notified nodes try to notify a neighboring unaware leader.

Finally, the rendezvous is concluded by the *notification step* S_3. A notified leader v attempts to notify all unaware nodes in $S(v)$ during this step. Note that this is the only time-interval during which an unaware non-leader node

Algorithm $\mathcal{A}_{clu}$: Code for leader v

upon wake-up do:

1: perform MIS algorithm of length $O(W + \log^2 n) \rightarrow$ become leader with cluster S(v)
2: choose rendezvous point r_1
3: $r_2 := r_1 + \eta\lceil \log n \rceil$, $r_3 := r_2 + (\gamma + \eta)\lceil \log^2 n \rceil$
4: **loop**
5: sleep or transmit with probability $\log n / W$ until next wake-up point r_1

S_1:
6: for $\eta \log n$ time slots transmit $\mathcal{M}_a(r_2)$ with probability $p_{MIS} \in \Theta(1)$

S_2:
7: for $(\gamma + \eta)(\lceil \log n \rceil + 1)^2 - \eta(\lceil \log n \rceil)$ time slots listen for $\mathcal{M}_n$
8: **if** $\mathcal{M}_n$ received **then**
9: transmit $\mathcal{M}_r$ for $\eta \log n$ time slots with probability $p_{MIS} \in \Theta(1)$
10: **end if**

11: sleep until r_3

S_3:
12: **if** notified **then**
13: for $\eta \log n$ time slots transmit $\mathcal{M}_n$ with probability $p_{MIS} \in \Theta(1)$
14: become non-leader
15: **end if**

16: $r_1 := r_1 + I$
17: $r_2 := r_1 + \eta\lceil \log n \rceil$
18: $r_3 := r_2 + (\gamma + \eta)\lceil \log^2 n \rceil$
19: **end loop**

must be awake. Summarizing, the actions during the rendezvous window are designed as to guarantee that a notification message in the neighborhood of a leader v is, first, passed to v, and second, passed from v to all nodes in $S(v)$. After the rendezvous window, a notified leader becomes a non-leader node in order to help informing other leaders located in its neighborhood. In the following, we give a more precise description of algorithm $\mathcal{A}_{clu}$ as performed by leaders and non-leaders, in which γ and η denote suitably large constants.

Consider a rendezvous window of leader v. In the *proclamation step* S_1, v transmits an announcement message $\mathcal{M}_a(r_2)$ containing the starting time of the second step of the rendezvous with a constant probability $p_{MIS} \in \Theta(1)$. Let u be a notified node with $(u, v) \in E$ and $u \notin S(v)$. Notified nodes remain listening in order to eavesdrop an announcement message of neighboring leaders. If node u receives such a message $\mathcal{M}_a(r_2)$ from v, it tries to notify v during the subsequent *leader-notification step*. In the analysis, we will show that with high probability *every* notified node in v's neighborhood will

receive $\mathcal{M}_a(r_2)$ from v.

In the *leader-notification step* S_2 all notified neighbors of v try to send a notification message $\mathcal{M}_n$ to v. Notice that if there are no notified neighbors of v, nothing happens during the *leader-notification step*. The procedure of informing a leader follows along the lines of the uniform algorithm presented in Section 14.2. Starting with probability $\frac{1}{2n}$, notified nodes exponentially increase their sending probability to speed up the notification. In order to prevent too much "noise" (i.e., too many nodes sending with high probability at the same time), v starts transmitting a reception message $\mathcal{M}_r$ with probability p_{MIS} as soon as it has received $\mathcal{M}_n$, thereby eventually stopping the increase in its cluster.

Finally, unaware nodes in $S(v)$ are only awake in the *notification step* S_3 starting from r_3. They are listening during these time slots, waiting for a possible notification message $\mathcal{M}_n$ from a potentially notified v.

Analysis Let $Q := \{s(u) \mid u \in \Gamma(v)\}$ be the set of all leaders that lead at least one node in v's neighborhood. In a doubling unit ball graph, it holds that $|Q| \leq \kappa_1$ for a constant κ_1. Also, let the constant κ_2 denote the maximal number of independent leaders in a node v's 2-hop neighborhood $\Gamma_2(v)$.

In the following, let $p_v(t)$ be the sending probability of node v in time slot t. Further, let $\Phi_v(t) := \sum_{u \in \Gamma(v) \setminus Q} p_u(t)$. In the next lemma, we show that given an upper bound on $\Phi_v(t)$, $\eta \log n$ time slots are sufficient to let a leader inform all its neighbors.

Lemma 14.3. *Let $v \in Q$ be a leader and consider a time interval $\mathcal{J}$ of length $\eta \log n$ during which v transmits with probability p_{MIS}. If $\Phi_v(t) \leq \chi$, for a constant $\chi \leq \frac{3\kappa_2}{2}$, then all nodes $w \in S(v)$ receive the message during $\mathcal{J}$ with probability $1 - 2n^{-3}$.*

Proof. As in the previous two chapters, we call a time slot *successful* if v transmits, but no other node in $\Gamma_2(v)$ transmits. In a successful time slot, all nodes in $\Gamma(v)$ receive the message from v without collision. The probability $P_{suc}(t)$ that a single time slot t is successful is at least

$$\begin{aligned} P_{suc}(t) &\geq p_{MIS} \cdot \prod_{\substack{w \in \Gamma_2(v) \\ w \neq v}} (1 - p_w(t)) \\ &\geq p_{MIS} \cdot (1 - p_{MIS})^{\kappa_2 - 1} \prod_{w \in \Gamma_2(v) \setminus Q} (1 - p_w(t)) \\ &\geq p_{MIS} \cdot (1 - p_{MIS})^{\kappa_2 - 1} \left(\frac{1}{4}\right)^{\kappa_2 \cdot \chi} \in \Theta(1). \end{aligned}$$

where the last inequality follows from Fact 2.1 and the assumption that $\Phi_v(t) \leq \chi$. Finally, the probability P_{no} that none of the $\eta \log n$ time slots is successful is bounded by

$$P_{no} \leq \left(1 - p_{MIS}(1 - p_{MIS})^{\kappa_2 - 1} \left(\frac{1}{4}\right)^{\kappa_2 \cdot \chi}\right)^{\eta \log n} \leq \frac{1}{2n^3}$$

for a suitably large constant η. □

Unfortunately, Lemma 14.3 holds only conditionally; based on the assumption that $\Phi_v(t) \leq \chi$. We now prove an upper bound on $\Phi_v(t)$ that holds throughout the execution of the algorithm with high probability.

Lemma 14.4. *With probability $1 - n^{-2}$, it holds for all t and for all leaders $v \in Q$ that $\Phi_v(t) \leq \chi$, where $\chi \leq \frac{3\kappa_2}{2}$ is a constant.*

Proof. At the beginning, when the first node is notified, the claim clearly holds. For the sake of contradiction, assume that the claim is first violated at leader v. Further, notice that Φ_v can only increase if some of its neighboring non-leader nodes are in the *leader-notification step* S_2. The idea is that as soon as v receives the notification message, it starts sending a reception message $\mathcal{M}_r$. We will show that the nodes in $\Gamma(v)$ receive this message and—by the definition of the algorithm—stop transmitting, which prevents Φ_v from increasing too much.

We define time slots t_v^* for a leader v, such that, $\Phi_v(t_v^*) < 1/2$ and $\Phi_v(t_v^* + 1) \geq 1/2$. By the same argument as in the proof of Theorem 14.2 (cf. Inequality (14.1)), we can bound $\Phi_v(t_v^* + (\gamma + \eta)(\lceil \log n \rceil + 1)) \leq 3/2$. That is, for all time slots t in the interval $\mathcal{J} = [t_v^* + 1, \ldots, t_v^* + (\gamma + \eta)(\lceil \log n \rceil + 1)]$, it holds that $1/2 < \Phi_v(t) \leq 3/2$. The probability $P_{suc}(t)$ that v receives a message without collision in an arbitrary time slot $t \in \mathcal{J}$ is at least

$$\begin{aligned} P_{suc}(t) &\geq \prod_{w \in Q \cap \Gamma(v)} (1 - p_w(t)) \cdot \sum_{w \in \Gamma(v) \setminus Q} \left(p_w(t) \cdot \prod_{\substack{q \in \Gamma(v) \setminus Q \\ q \neq w}} (1 - p_q(t)) \right) \\ &\geq (1 - p_{MIS})^{\kappa_2} \cdot \Phi_w(t) \left(\frac{1}{4}\right)^{\Phi_w(t)} \\ &\geq (1 - p_{MIS})^{\kappa_2} \cdot \frac{3}{2} \left(\frac{1}{4}\right)^{3/2} > \frac{(1 - p_{MIS})^{\kappa_2}}{6}. \end{aligned}$$

The first $\gamma(\lceil \log n \rceil + 1)$ time slots of $\mathcal{J}$ suffice such that v receives $\mathcal{M}_n$. Specifically, the probability P_{no} that none of these time slots is successful is

$$P_{no} \leq \left(1 - \frac{(1 - p_{MIS})^{\kappa_2}}{6}\right)^{\gamma(\lceil \log n \rceil + 1)}, \tag{14.2}$$

which again can be made $P_{no} \leq n^{-3}/2$ for large enough constants γ. Once, node v receives $\mathcal{M}_n$, it will try to acknowledge by sending $\mathcal{M}_r$. By the assumption that v is the *first* leader to violate the claim, we know that there are at least $\eta(\lceil \log n \rceil + 1)$ time slots in $\mathcal{J}$ remaining during which $1/2 < \Phi_v(t) \leq 3/2$ holds. Consequently, we can apply Lemma 14.3, that is, with probability at least $1 - n^{-3}$, the message $\mathcal{M}_r$ will be received by all nodes in $\Gamma(v)$ within the $\eta(\lceil \log n \rceil + 1)$ time slots. Hence, the probability that v is the first node to violate the claim is bounded by $2 \cdot n^{-3}/2$ for

suitably large constants γ and η. Because there are at most n leaders in the network and every leader needs to be notified only once, the Lemma holds with probability $1 - n^{-2}$. □

The following Corollary is implicit in the proof of Lemma 14.4 (cf. Inequality (14.2)).

Corollary 14.5. *Consider a leader $v \in Q$ and the leader-notification step S_2 of a notification window. If there exists a notified node in $\Gamma(v) \setminus Q$, v will be notified at the end of S_2.*

Based on Lemma 14.4, we can now state the main theorem of correctness.

Theorem 14.6. *With probability at least $1 - 1/n$, the algorithm works as demanded: each leader v successfully announces to all its neighbors about the proclamation step S_1 for the entire duration of the notification phase. As soon as there exists a notified non-leader in $\Gamma(v)$, v will be notified in the following leader-notification step S_2. And finally, a notified leader v will inform all its neighbors $u \in \Gamma(v)$ in the notification-step S_3 following v's notification.*

Proof. The steps S_2 and S_3 follow directly from Lemma 14.3, Corollary 14.5, and the fact that there are at most n leaders, each of which is notified at most once. By Lemma 14.3, every attempt of sending a $\mathcal{M}_a$ message is successful with probability $1 - n^{-3}$. Each of the n nodes needs to send at most n messages $\mathcal{M}_a$ during the notification phase. The proof is concluded because the set of leaders is connected if we consider all two-hop paths in G. □

Of particular interest is the energy efficiency and its comparison to the two previous algorithms. Let $m \leq n$ be the number of leader nodes in the network and let ξ denote the energy efficiency $E(\mathcal{A}_{clu}, T_D)$. Clearly, the ratio m/n depends on the *density* of the network. The following theorem quantifies the achieved trade-off.

Theorem 14.7. *Let $f(n)$ be a function such that algorithm $\mathcal{A}_{clu}$ has time efficiency $T(\mathcal{A}_{clu}, T_D) \leq f(n)$. Let m be the number of* leaders *chosen by $\mathcal{A}_{clu}$. In a unit disk graph G with high node density and for a given T_D, $\mathcal{A}_{clu}$'s energy efficiency $\xi = E(\mathcal{A}_{clu}, T_D)$ is bounded by*

$$\xi \in O\left(\frac{\frac{f(n)}{n \log n} + \log^2 n}{T_D} + \frac{n \log n}{f(n)} + \frac{m \log^2 n}{f(n)}\right).$$

Proof. The choice of I's length determines the trade-off between energy-efficiency and the speed of notification. We have to choose I such that with high probability, the notification broadcast is finished within time $f(n)$. We do so by setting I to a value guaranteeing that the notification proceeds at least one hop in time $f(n)/n$ with high probability. That is, we set $I := \lceil f(n)/n \rceil$.

Upon waking up, each node v has initialization costs $e_{init}(v) \in O(W + \log^2 n)$. For the maintenance costs during the deployment phase, we distinguish between leaders and non-leader nodes. Non-leaders are awake for the

duration of $\eta \log n$ during each rendezvous interval of length I. Thus, for non-leaders, $a(v) = \eta\lceil \log n \rceil / I$. Leaders must be awake longer in each rendezvous interval, namely $2\eta \log n + (\gamma+\eta)(\lceil \log n \rceil + 1)^2 \in O(\log^2 n)$ time slots. Additionally, leaders need to transmit with probability $\log n / W$ in each time slot.

Therefore, for appropriate constants α, $\delta > \gamma + \eta$, and η the energy efficiency $E(\mathcal{A}_{clu}, T_D)$ of $\mathcal{A}_{clu}$ is at most

$$\begin{aligned} \xi &= \frac{1}{nT_D}\left[\sum_{v\in Q} \alpha\left(W + \log^2 n\right) + T_D \sum_{v\in Q}\left(\frac{\log n}{W} + \frac{\delta \log^2 n}{I}\right)\right. \\ &\quad \left. + \sum_{v\in V\setminus Q}\left(\alpha(W+\log^2 n) + \frac{T_D \eta \log n}{I}\right)\right]. \end{aligned}$$

Setting $I = \lceil f(n)/n \rceil$ and $W = I/\log n$, we obtain

$$\begin{aligned} \xi &= \frac{\alpha(W+\log^2 n)}{T_D} + \frac{m}{n}\left(\frac{\log n}{W} + \frac{\delta \log^2 n}{I}\right) + \frac{n-m}{n}\cdot\frac{\eta \log n}{I} \\ &\leq \frac{\alpha\left(\frac{f(n)}{n\log n} + \log^2 n\right)}{T_D} + \frac{m}{n}\cdot\frac{(\delta+1)n\log^2 n}{f(n)} + \frac{n-m}{n}\cdot\frac{\eta n \log n}{f(n)} \\ &\in O\left(\frac{\frac{f(n)}{n\log n} + \log^2 n}{T_D} + \frac{m\log^2 n}{f(n)} + \frac{n\log n}{f(n)}\right). \end{aligned}$$

□

Observe that the first asymptotic term of Theorem 14.7 contains T_D in the denominator. This captures the fact that the amount of energy spent on initializing a structure weighs more or less heavily, depending on the respective length of the deployment phase. Specifically, this term can be neglected if the deployment phase is long. As for the two remaining terms, they express the energy efficiency of leaders and non-leaders, respectively.

Discussion

This section discusses the results obtained in Theorems 14.1, 14.2, and 14.7. For the comparison, we demand all three algorithms to finish their notification phase within a fixed amount of time $f(n) \in \Theta(n^2)$, $f(n)$ being the same for all algorithms. This allows us to compare the energy efficiency $E(\mathcal{A}, T_D)$ each algorithm is required to invest in order to ensure that the notification is finished within time $f(n)$. As mentioned in Section 14.1, we obtain the same results when asking the question the other way around, i.e., when fixing the algorithm's energy efficiency and comparing the resulting time efficiency $T(\mathcal{A}, T_D) = f(n)$. Table 14.1 shows the results derived from Theorems 14.1,

14.2, and 14.7 under the assumption that the length of the deployment phase T_D is long enough compared to $f(n)$[2].

Algorithm $\mathcal{A}$	Energy Efficiency $E(\mathcal{A}, T_D)$
$\mathcal{A}_{birth}$ [174]	$\Theta(1)$
$\mathcal{A}_{uni}$	$\Theta(\frac{\log^2 n}{n})$
$\mathcal{A}_{clu}$	$\Theta(\frac{\log n}{n} + \frac{m \log^2 n}{n^2})$

Table 14.1: A comparison of the energy efficiency of the three algorithms for a fixed $T(\mathcal{A}, T_D) = f(n) \in \Theta(n^2)$ and large enough T_D.

It is clear that both $\mathcal{A}_{clu}$ and $\mathcal{A}_{uni}$ significantly outperform $\mathcal{A}_{birth}$, regardless of the network density or, generally, the ratio between leaders vs. non-leaders. It is interesting to study the relative strengths of $\mathcal{A}_{clu}$ and $\mathcal{A}_{uni}$. Asymptotically, the trade-off achieved by $\mathcal{A}_{clu}$ is strictly better than $\mathcal{A}_{uni}$ if $m \in o(n)$, that is, if less than a constant fraction of the nodes are leaders. If, for instance, $m \in O(n/\log n)$, the resulting asymptotic energy-efficiency is $E(\mathcal{A}_{clu}, T_D) \in O(n \log n / f(n))$, which is better than $\mathcal{A}_{uni}$ by an $O(\log n)$ factor. In case the number of leaders is a constant fraction of n, the asymptotic energy efficiency is $O(n \log^2 n / f(n))$, which equals the trade-off achieved by $\mathcal{A}_{uni}$. Hence, depending on the *network density* and the resulting number of leaders, the asymptotic energy efficiency of $\mathcal{A}_{clu}$ is either better than or equal to that of $\mathcal{A}_{uni}$. That is, the higher the network density, the more worthwhile it becomes to invest initial energy on obtaining a cluster-based semi-structure.

[2]Note that $f(n) \in \Theta(n^2)$ is the smallest value for $f(n)$ so that $\mathcal{A}_{birth}$ is capable of finishing its notification phase within $f(n)$ in arbitrary networks. Similar results as shown in Table 14.1 can be obtained for higher values of $f(n)$ in a straightforward way.

Chapter 15

Conclusions and Outlook

Part II of this thesis studies the impact of the harsh physical conditions in wireless ad hoc and sensor networks on the complexity of distributed coordination tasks. Focusing particularly on the deployment and initialization phase of these networks—when nodes may wake up asynchronously without any knowledge about the network—we have devised efficient algorithms for the important network coordination primitives colorings (Chapter 12) and clustering (Chapter 13). Energy-efficiency is one of the most crucial aspects in the design of *wireless sensor networks*. Based on the MIS algorithm, Chapter 14 presented a method for improving the delay-energy trade-off during the deployment of networks with high density. There appear to be applications for our algorithms in unstructured radio networks even beyond the deployment problem. In [85], for instance, a protocol is presented which—based on the MIS algorithm of Chapter 13—constructs an initial infrastructure in wireless multi-hop network. It is conceivable, that the algorithm finds further application in the future.

A common complaint within the networking community in general, and the ad hoc and sensor networking community in particular, is that there exists a wide chasm between theory and practice. In this sense, it is important and interesting to study algorithms for wireless ad hoc and sensor networks in models that capture more closely the harsh conditions under which they ultimately have to operate. The unstructured radio network model makes a step towards this direction, but remains concise enough to allow for stringent, algorithmic analysis.

From a more theoretical point of view, numerous interesting questions remain unresolved. Besides devising efficient algorithms for other problems in the unstructured ratio network model, one of the foremost question is the relative capabilities of *deterministic* versus *randomized algorithms*. In [30], Bar-Yehuda, Goldreich and Itai established an exponential gap between deterministic and randomized algorithms for the broadcast problems in constant-diameter networks. Ever since, there has been an intensive effort to determine the exact deterministic time complexity of communication primitives

such as broadcasting or wake-up, e.g. [48, 49, 52, 63, 104, 128]. On the other hand, nothing is known about the deterministic distributed complexity of computing network coordination structures such as dominating sets, maximal independent sets, or colorings in the unstructured radio network model, and it would be interesting to further investigate this direction. In particular, it is presumable that the algebraic structures obtained in the study of the wake-up problem (radio-synchronizers and universal synchronizers) may turn out to be useful tools for developing efficient deterministic solutions for coloring or MIS in the unstructured radio network model. On the other hand, the differences between the notion of asynchronous wake-up as imposed by the unstructured radio network model and work on the wake-up problem, respectively, may need to be addressed using novel techniques.

Another question that we have left unexplored is the value of having a collision detection mechanism. Throughout Part II of this thesis, we have assumed that nodes are unable to distinguish a collision from the case in which no message arrives. But, what if radio modules are enhanced with a collision detection mechanism that can give a ternary feedback (collision, successful reception, or free medium), instead of a binary one? The $\Omega(\log^2 n)$ time lower bound for clear radio transmission in single-hop networks has been derived only at the assumption that nodes have no collision detection mechanism [86]. Moreover, the lower bound construction of [86] does not seem to be easily extendible to the collision detection case. This raises hopes that by making use of collision detection in a clever way, there may be an MIS algorithm in the unstructured radio network with a better running time than $O(\log^2 n)$. Inspecting our MIS algorithm reveals that the availability of a collision detection mechanism cannot help improving its running time. Hence, any algorithm that—based on the ability to detect collisions—achieves a running time of $o(\log^2 n)$ must probably be based on a different algorithmic approach.

It is interesting to relate our results of Part II of this thesis to their respective counterparts in the synchronous message passing model established in Part I. In particular, in combination with the $\Omega(\log^2 n)$ lower bound, our algorithm in Chapter 13 settles the exact (randomized) time complexity for computing an MIS in the unstructured radio network as $\Theta(\log^2 n)$. This is by a logarithmic factor slower than Luby's MIS algorithm for general graphs in the synchronous message passing model. Moreover, we have shown in Chapter 8 that in the message passing model, even more efficient solutions are possible for the MIS problem in graphs with bounded independence. Specifically, in the $\mathcal{CONGEST}$ model, an MIS can be computed in time $O(\log \Delta \log^* n)$. In the $\mathcal{LOCAL}$ model with known distances, even $O(\log^* n)$ communication rounds suffice for the computation of an MIS, which is asymptotically optimal. These results (almost precisely) quantity the loss of computational efficiency caused by the harsh characteristics of the unstructured radio network model as opposed to the synchronous message passing model. This allows for an interesting comparison between these two models of distributed computing.

Part III

Scheduling Complexity of Wireless Networks

Chapter 16

Wireless Networks Beyond Graph Models

Throughout Part I and II of this thesis, we have modeled wireless multi-hop networks such as sensor, ad hoc, or mesh networks by means of *graphs*. In fact, studying such networks on the abstraction layer of graphs has become a quasi-standard in most of the *algorithmic* research in this field. In its most general form, a graph model for wireless networks consists of two graphs: A *connectivity graph* $G_c = (V, E_c)$ and an *interference graph* $G_i = (V, E_i)$. Both graphs are based on the set of devices V. A receiver v successfully decodes a message from a sender u, if and only if u and v are neighbors in the connectivity graph, $(u, v) \in E_c$, and v does not have a concurrently transmitting neighbor node in the interference graph G_i. In graph-based models, a protocol designer must therefore take care that no neighbor in G_i is transmitting simultaneously to a neighbor in G_c, or at least, that this happens rarely. It is therefore not surprising that in graph theory, interference-free concurrent transmissions boil down to solving variants of independent set or coloring problems (e.g. [205]).

It is clear that when modeling the wireless communication medium, the concept of an *edge* is a stark oversimplification, as it is a binary representation for continuous (non-binary!) physicals laws. In fact, a node barely outside the interference range of a receiver v (that is, a node not connected by an edge with v in G_i) might still cause enough interference such that receiver v is not able to decode a message from a legitimate neighboring sender in G_c. Moreover, the interference of several senders can cumulate and cause a collision at an intended receiver, even though the interference generated by each individual sender may be harmless by itself.

While in algorithmic research on wireless multi-hop networks, graph models have been studied almost exclusively, communication theorists have typically trusted graph-based models much less. In communication or information theory, researchers are studying an arsenal of fading channel models, the

simplest being the signal-to-interference-plus-noise ratio (SINR) model. In the SINR model, the energy of a signal fades with the distance to the power of the path-loss parameter α. If the signal strength received by a device divided by the interfering strength of competitor transmitters (plus the noise) is above some threshold β, the receiver can decode the message, otherwise it cannot. This simple SINR model is unrealistic as well, mostly because it is inherently geometric: In reality antennas are not perfectly isotropic, and even more importantly the environment is obstructed by walls or plants that shield or reflect the signal. Although some of these issues can be integrated into the basic SINR model, these "SINR+" models are predisposed to become complicated – essentially intractable from the point of view of a protocol designer. Graph-based models such as the bounded-independence graph (BIG) introduced in Chapter 8, on the other hand, are capable of automatically incorporating both imperfect (or even directional) antennas and terrains with obstructions. As a consequence, SINR models have been considered almost exclusively by communication and information theorists when studying fundamental scaling laws (e.g. the theoretical capacity of wireless multi-hop networks [117]).

From an algorithmic point of view, however, SINR models have been widely accepted as being too low-level and too intricate for thoroughly comprehending protocols, let alone analytically prove their correctness and/or efficiency. Also, because of their cleaner abstraction layer, it seems that a majority of classes, books, or tutorials therefore prefers to teach the higher-layer algorithmic basics in wireless multi-hop networking in terms of graphs, not in terms of physical SINR models.

As we will see in Chapters 17 and 18, there exist important physical phenomena which graph-based models inherently fail to capture. This fact by itself is of course neither new nor surprising (see for instance [33, 114]). The more interesting question is whether the resulting inconsistencies are small enough to be justified by the gained simplicity of the model. Or in other words, the question is how drastic a change of the nature of wireless communication is caused by exactly those physical phenomena that *cannot* be modeled by graphs. In particular, do these inconsistencies render theoretical boundaries and analytical proofs in graph-based models entirely inaccurate, thus undermining their significance?

More specifically, we raise the question whether there are important applications or tasks in wireless multi-hop networks in which provably efficient (possibly even theoretically optimal) graph-based algorithms perform inherently worse than algorithms that are designed to make use of SINR aspects. If there were no such examples, it would serve as a major justification for studying networks on the clean abstraction layer of graphs. If, however, there are examples in which the performance of any graph-based algorithm is surpassed by algorithms explicitly taking SINR into account, it would highlight the need for a more physical-level oriented design and analysis of network protocols.

In Part III of the thesis, we show that the inconsistencies created by graph-based models can indeed be overwhelming, even for simple scheduling tasks. In particular, scheduling algorithms and MAC layer protocols whose design

has been guided with graph-models in mind can perform sub-optimally as compared to protocols that are explicitly defined for the SINR model. Specifically, Chapter 17 exemplifies the difference between graph-based models and SINR models. By means of simple calculations and practical measurements using standard sensor nodes, it is shown how graph-based models are unable to properly model simple scheduling problems in wireless networks.

These observations lead to interesting questions regarding the fundamental laws that governs scheduling in wireless networks. In particular, consider a set of communication requests that need to be scheduled over a wireless medium. In one time slot, only a subset of all the desired communication links can be scheduled in parallel; and in every subsequent time slot, a subset of the remaining unscheduled links may be scheduled, until finally all links are scheduled. Against this background, it is clearly advantageous to find a short schedule, ideally one with minimal length. This allows for higher bandwidth and, ultimately, higher throughput. In Section 17.3, we define the *scheduling complexity* as a measure for describing the minimal amount of time required to *physically* establish a set of communication requests.

In the subsequent chapters, we study the scheduling complexity of various request patterns in the physical SINR model of wireless propagation. Chapter 18 shows that simple and intuitive scheduling and power assignment schemes inevitably result in highly sub-optimal schedules, even for simple request sequences. As a corollary, these results imply that the throughput achieved by all currently employed MAC layer protocols for wireless multihop networks can be by a factor of $\Omega(n)$ worse than the optimum.

Chapter 19 then shows that the inefficiency of the above MAC layer protocols is not unavoidable. In particular, we propose an algorithm that constructs a spanning tree, and assigns power levels and time slots to each link of the tree such that in polylogarithmic time, all transmissions are received correctly, i.e., without violating the signal-to-interference plus noise ratio at any receiver. Our algorithm implies that even in worse-case networks, the scheduling complexity of a strongly-connected topology is polylogarithmic in n and such topologies can thus be scheduled efficiently. As shown in [117], the theoretically achievable *capacity* of wireless networks increases asymptotically as $\Theta(\sqrt{n})$, which indicates a fundamental scaling problem in large-scale wireless multi-hop networks. In contrast, our results show that there is no comparable scalability problem when it comes to *scheduling* certain communication requests. The polylogarithmic scheduling algorithm further shows that the remedy against slow scheduling is a highly non-linear assignment of power levels. Particularly, *many different power levels* are required in order to achieve an efficient schedule.

For more general topologies or communication requests, it may not be insightful to bound the scheduling complexity as a function of n. Chapter 20 therefore derives the scheduling complexity of *arbitrary communication requests* in terms of static, graph-based interference measures, thus establishing—at the end of Part III—a relationship between SINR models and graph-based models.

Chapter 17

Models and Definitions

This chapter formally introduces the Physical model of wireless communication that is based on an explicit consideration of the *Signal-to-Interference-plus-Noise-Ratio* (SINR).

17.1 SINR: Modeling Interference

We model a wireless network as a set of nodes $V = \{v_1, \ldots, v_n\}$ that are arbitrarily located in the Euclidean plane. The Euclidean distance between two nodes $v_i, v_j \in V$, is denoted by $d(v_i, v_j)$. The *ball* $B(v_i, r)$ of radius r around node v_i contains all nodes $v_j \in V$ for which $d(v_i, v_j) \leq r$. For simplicity and without loss of generalization, we assume that the minimal distance between any two nodes is 1.

The core aspect of the communication model underlying our analysis is the description of the circumstances under which a message is correctly received by its intended recipient. In the *Signal-to-Interference-plus-Noise-Ratio* (SINR) model (also called *Physical Model* in [117]), the successful reception of a transmission depends on the received signal strength, the interference caused by simultaneously transmitting nodes, and the ambient noise level. Let P_r be the received power of a signal sent to a node v_r, and denote by I_r the interference power generated by other nodes in the network. Finally, let N be the ambient noise power level. Then, a node v_r receives a transmission if and only if $\frac{P_r}{N+I_r} \geq \beta$, where β is the minimum signal-to-interference-ratio that is required for a message to be successfully received at v_r.[1]

In wireless networks, the value of the received signal power P_r of a signal is a decreasing function of the distance $d(v_s, v_r)$ between the transmitter v_s and the receiver v_r. Theoretically, the received signal power P_r can be

[1] All results in Part III can be generalized such that every node v_i has its own β_i.

modeled as decaying with distance $d(v_s, v_r)$ as

$$P_r \;=\; \frac{P_s}{d(v_s, v_r)^\alpha},$$

where P_s is the sending power of the transmitting node. The so-called *path-loss exponent* α is a constant between 2 and 6, whose exact value depends on external conditions of the medium (humidity, obstacles, ...), as well as the exact sender-receiver distance. As common, we assume that $\alpha > 2$ [117].

As for the notation in this paper, we occasionally use the formulation $I_r(v_i) = P_r(v_i)$ in order to emphasize that the signal power transmitted by a node v_i other than the intending sender is perceived at v_r as interference. In summary, if $P_r(v_s)$ and $I_r(v_i)$ are the received power levels sensed by node v_r in a specific time slot, a signal transmitted by a node $v_s \in V$ is successfully received by v_r if

$$\frac{P_r(v_s)}{N + \sum_{v_i \in V \setminus \{v_s\}} I_r(v_i)} \;=\; \frac{\frac{P_s(v_s)}{d(v_s, v_r)^\alpha}}{N + \sum_{v_i \in V \setminus \{v_s\}} \frac{P_s(v_i)}{d(v_i, v_r)^\alpha}} \;\geq\; \beta. \qquad (17.1)$$

Finally, the *total interference* I_r experienced by a receiver v_r is the sum of the interference power values created by all nodes in the network (except the intending sender v_s), that is, $I_r := \sum_{v_i \in V \setminus \{v_s\}} I_r(v_i)$.

Generalized Physical Model

In practice, the received signal power may deviate from the above theoretical bound for various reasons. On the one hand, the signal-emitting characteristics of antennas may not be perfectly omni-directional. Moreover, shadowing, reflection, scattering, and diffraction caused by the presence of obstacles to wireless signal propagation may have an impact on the signal power actually sensed at the receiver.

In order to better account for some of these aspects of wireless communication, we define and study the following slight generalization of the physical model, which we call the *generalized physical model*. In this generalized physical model with parameter θ, the received signal power (as well as the interference caused by simultaneously transmitting nodes) can deviate arbitrarily from the theoretically received power by a factor of θ. Formally, if $P_r(v_s)$ is defined to be the actual received power of a signal transmitted by node v_s as sensed by the receiving node v_r, the generalized physical model states that $P_r(v_s)$ is in the range

$$\frac{1}{\theta} \cdot \frac{P_s}{d(v_s, v_r)^\alpha} \;\leq\; P_r(v_s) \;\leq\; \theta \cdot \frac{P_s}{d(v_s, v_r)^\alpha}.$$

Note that the model leaves open the exact received signal power, and hence algorithms working in the generalized physical model must be robust enough to cope with arbitrary (even worst-case) deviations within the stated bounds. Clearly, for $\theta = 1$, the generalized physical model is equivalent to the standard physical model.

17.2 Graphs vs. SINR: Simple Examples

In order to exemplify the difference between graph models and SINR models, consider the simple network with 4 nodes illustrated in Figure 17.1. As indicated by the arrows, nodes v_1 and v_3 want to send a message to the corresponding receivers v_2 and v_4, respectively. A typical graph-based communication model implies that scheduling both requests—regardless of the transmission powers of the senders—requires at least two time slots, because when being transmitted simultaneously, the two messages would collide at v_2. This is a classic example of the so-called Hidden-Terminal Problem.

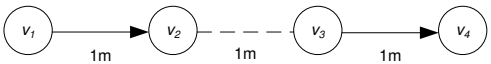

Figure 17.1: Four nodes placed equidistantly in a line.

Next, consider the example network of Figure 17.2, in which the two sender-receiver pairs (v_1, v_2) and (v_3, v_4) are rearranging in such a way that one pair is placed in the transmission line of the other. Node v_4 being inside the *transmission range* of sender v_1, graph-based models again imply that scheduling both messages simultaneously is impossible due to interference (and a resulting collision) at v_4.

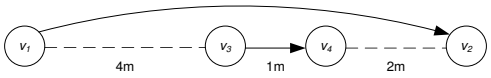

Figure 17.2: A more elaborate example with four nodes.

While graph-based models thus imply that simultaneous scheduling of messages in the examples of Figures 17.1 and 17.2 is impossible, the reality looks very different. In fact, the two requests can easily be transmitted in parallel in both settings!

Using the physical SINR model, the feasibility of simultaneous transmission in the example networks of Figures 17.1 and 17.2 can be derived analytically. For a simple calculation, assume $\alpha = 3$, $\beta = 3$, and background noise $N = 10nW$. Those values are realistic, even pessimistic, for sensor networks [218]. In the example of Figure 17.1, let v_1 and v_3 send with power $P_{v_1} = 0dBm$[2] and $P_{v_3} = -7dBm$, respectively. We get the following SINR values: $\beta_{v_2} = \frac{1000\mu W/(1m)^3}{0.01\mu W+(200\mu W/(1m)^3)} \approx 5.00$ and $\beta_{v_4} = \frac{200\mu W/(1m)^3}{0.01\mu W+(1000\mu W/(3m)^3)} \approx 5.40$. Consequently, both receivers can correctly decode their corresponding message without problems. In the network shown in Figure 17.2, too, both messages can be transmitted simultaneously. When

[2]The unit dBm is an abbreviation for the power ratio in decibel (dB) of the measured power referenced to one milliwatt (mW). Zero dBm equals one milliwatt and in general, $x = 10\log_{10}(P/1mW)$, where x and P are measured in dBm and mW, respectively.

v_1 transmits with $P_{v_1} = 1dBm$ and v_3 with $P_{v_3} = -15dBm$, the SINR at v_2 and v_4 amounts to $\beta_{v_2} = 3.11$ and $\beta_{v_4} = 3.13$, respectively.

Practical Measurements

This section shows that the effects described above are not merely theoretical gimmick, but can be verified using widely available standard sensor nodes. Specifically, we employed mica2 sensor nodes running with TinyOS [123]. Each node is equipped with a 7.38 MHz Atmel processor, 128 kB of program memory, and a ChipCon CC1000 radio transceiver configured to send at a frequency of 868 MHz. The testbed consisted of two senders v_1 and v_3 and two corresponding receivers v_2 and v_4 positioned in a line similar to the setup shown in Figure 17.2. The distances between neighboring nodes were scaled down to $100cm$, $30cm$, and $60cm$, respectively.

On the sender side, the application tries to send 20000 messages in succession to the corresponding receiver. The receiver application simply counts the number of messages received. For the success of this experiment, it was crucial that the MAC layer allows parallel transmission of multiple messages. Unfortunately, this is not possible with the built-in MAC layers available for TinyOS because they are designed for a graph-based representation of sensor networks and therefore try to prevent collisions. Consequently, the default MAC layer used by TinyOS was replaced by an "SINR-MAC", which we tailored for our experiments:

- *Sniffing the medium:* Before sending a message, no check is performed whether the medium is free. This allows simultaneous transmission of several messages.
- *Initial back-off:* The initial back-off before transmitting a message is removed. The default carrier sensing MAC layer protocol uses this random delay to prevent simultaneous start of transmission by multiple sensors.
- *Output power:* The sender v_1 sends with output power $0dBm$, v_3 sends with $-10dBm$.

Before presenting the measurement results, we calculate a theoretical lower bound for the time required to transmit the 20000 messages when assuming a graph-based communication model. A single packet consists of a preamble (8 bytes), two synchronization bytes, a header of size 7 bytes (destination address (2), message type (1), group ID (1), payload length (1), and the CRC (2)), and the actual payload which is set to 6 bytes throughout the experiments in this section. The sensors transmit with $38.4kbps$ using Manchester Encoding which results in a data rate of $19.2kbps$ ($2.4kBps$). In addition to the transmission, the radio module must switch from RX to TX mode and back to RX mode for each message. This takes about $0.5ms$ according to [44]. Summed up, at least $(23bytes/2.4kBps + 0.5ms) * 40000packets) \approx 403s$ are required when assuming a graph-based model in which parallel transmission

is not possible. Note that this is a lower bound ignoring any software overhead, e.g. for copying the message to the corresponding memory location. Therefore, even stronger lower bounds could be proven with a more sophisticated calculation.

The table below shows the average results of multiple runs of the same experiment. The left column contains the measured values when using the default TinyOS MAC layer protocol. On the right, the results of our adjusted "SINR-MAC" are displayed.

	Time required	
	standard MAC	"SINR-MAC"
Node v_1	$603s$	$267s$
Node v_3	$591s$	$268s$

	Messages received	
	standard MAC	"SINR-MAC"
Node v_2	19998	18668
Node v_4	18852	19916

These results show that the examples analyzed in the previous section can be implemented in practice. On the one hand, the time used by the default MAC layer protocol exceeds the calculated lower bound by almost 50%. On the other hand, the "SINR-MAC" exploiting the interference phenomena of the SINR model performs significantly better than this limit. This highlights the inherent inability of graph-based models to represent certain important physical aspects of wireless networks. More importantly, the experiment indicates that protocols explicitly tailored for the SINR model—in this case the adjusted "SINR-MAC" layer protocol—can often outperform conventional graph-based protocols.

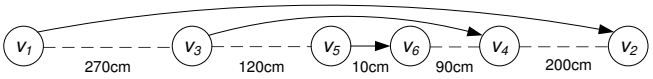

Figure 17.3: Three interleaved sender-receiver pairs.

The question arises whether it is practically feasible to have three, four or more senders transmitting messages in parallel when they are located in a line. One constraint that ultimately limits the number of multiple simultaneous transmissions is the interval of possible output power levels of the radio module. For example, the ChipCon CC1000 module integrated in the mica2 sensors only allows power levels between $-20dBm$ and $+5dBm$ at a frequency of $868MHz$. Additionally, not every value in the output power interval may be assigned. Our experiments indicated that—while still being practically feasible—deploying multiple pairs of nodes in a line similar to Figure 17.3 becomes increasingly difficult and failure-prone as the number of pairs increases. However, using alternative sensor platforms with different radio modules, it might be feasible to have more than three senders sharing the medium in parallel.

17.3 The Scheduling Complexity

If too many devices transmit simultaneously in a wireless network, the interference caused by these transmissions will prevent an intended receiver from receiving the signal, i.e., the message is lost. On the other hand, if too few nodes transmit at the same time, valuable bandwidth is wasted and the overall throughput may suffer. Hence, the classic problem faced by any MAC layer or scheduling protocol is that neither selecting too many nor too few devices for concurrent transmission is acceptable.

In order to gain a more thorough understanding of the performance of MAC layer protocols and in order to learn about the fundamental possibilities and limitation of "optimal" scheduling protocols, we need a measure that captures the achievable equilibrium between interference and simultaneous transmissions. More concretely, assume that we are given a set of directed links between pairs of nodes that indicate communication requests. How much *time* is required to schedule all these requests in a wireless multi-hop network? In the sequel, we define and study this *scheduling complexity* in wireless networks. As we have seen in the examples of Section 17.2, understanding the scheduling complexity can yield intriguing insights into the algorithmic structure of wireless communication beyond graph-based models.

Formally, we assume—as in Part II of this thesis—that transmissions are slotted into synchronized slots of equal length. In each time slot t, a node v can either transmit or not transmit. If it transmits, it chooses a power level $P_v > 0$ that must be sufficiently large in order to reach the intended receiver. A *power assignment* determines the power level chosen by each node in a certain time slot. Formally, a power assignment ϕ_t is a function $\phi_t : V \mapsto \mathbb{R}^+$ which maps every node in the network to a power level. We denote by $\phi_t(v_i)$ the power level of node $v_i \in V$ in time slot t. If a node is not scheduled to transmit in this time slot, then $\phi_t(v_i) = 0$. In case it is clear from the context which time slot t is meant, we also use the notational short-cut $P_i := \phi_t(v_i)$. A *schedule* $\mathcal{S} = (\phi_1, \ldots, \phi_{T(\mathcal{S})})$ is a sequence of $T(\mathcal{S})$ power assignments, where ϕ_t denotes the power assignment in time slot t. Finally, we call $T(\mathcal{S})$ the *length* of schedule $\mathcal{S}$. That is, a schedule $\mathcal{S}$ of length $T(\mathcal{S})$ determines the power level P_i for every node $v_i \in V$ for $T(\mathcal{S})$ consecutive time slots.

Let Λ be a set of *communication requests* λ_{ij}. Each request λ_{ij} denotes a directed link (v_i, v_j) and indicates that node v_i is supposed to successfully transmit a message to node v_j. The task of a scheduling algorithm is to schedule a set of communication requests Λ such that all messages are *successfully* received.

Definition 17.1. *Consider a time slot t and a power assignment ϕ_t. We say that a directed link (v_i, v_j) is* successfully scheduled *in time slot t if v_j successfully receives a message from v_i according to the SINR Inequality* (17.1).

Let L_t be the set of all successfully scheduled links in time slot t. The goal is that after as few time slots as possible the union of all sets L_t equals the set of requests Λ.

Definition 17.2. *Let Λ be the set of communication requests. The scheduling problem for Λ is to find a schedule $\mathcal{S}$ of minimal length $T(\mathcal{S})$ such that the union of all successfully transmitted links $\bigcup_{t=1}^{T(\mathcal{S})} L_t$ equals Λ.*

Finally, we define the *scheduling complexity* of a topology, that is, of an arbitrary set of communication requests Λ.

Definition 17.3. *The* scheduling complexity $T(\Lambda)$ *of a set of communication requests Λ is the minimal number of time slots T such that there exists a valid schedule $\mathcal{S}$ of length $T = T(\mathcal{S})$.*

The scheduling complexity $T(\Lambda)$ reflects how fast all requests in Λ can theoretically be satisfied (that is, when scheduled by an optimal MAC layer protocol).

Often, as in Chapters 18 and 19, we will be interested in the scheduling complexity of a certain network property Ψ, rather than a specific set of communication requests Λ. By a network property Ψ, we mean a desirable network topology such as a strongly-connected subgraph, a spanner, a low-degree topology, etc., depending on the specific requirements at hand.

Definition 17.4. *The* scheduling complexity *of a network property Ψ is the minimal number of time slots T, such that there exists a valid schedule $\mathcal{S}$ of length $T = T(\mathcal{S})$, such that the union of all successfully transmitted links $\Sigma = \bigcup_{t=1}^{T(\mathcal{S})} L_t$ satisfies property Ψ.*

For instance, the *scheduling complexity of strong-connectivity* (i.e., Ψ is strong-connectivity) translates to finding a schedule $\mathcal{S}$ of minimal length in which all successfully transmitted links connect the network, i.e., there exists a path between all pairs of nodes. Finally, when talking about a specific scheduling algorithm $\mathcal{A}$, we refer to the *scheduling complexity of $\mathcal{A}$* as the number of time slots required by this algorithm in the worst-case to schedule a specific set of requests.

Understanding the scheduling complexity of various network properties as well as arbitrary sets of requests is of fundamental interest in wireless networks. It is a measure that indicates how quickly a desired network topology can actually be established or how quickly communication requests can be satisfied. In this sense, studying the scheduling complexity complements the study of *capacity* in wireless networks [117]. While the notion of *capacity* captures the amount of information that can be transmitted in a *best-case* or *average-case* scenario (i.e., without assuming worst-case networks), the *scheduling-complexity* of a wireless network describes how quickly information can be transmitted in *worst-case* scenarios, i.e., in arbitrary networks. Moreover, note that the scheduling complexity of a set of requests places a lower bound on the theoretically achievable efficiency of any MAC layer or scheduling protocol. As such, this measure provides a handle for theoretically analyzing the performance of MAC layer protocols from an algorithmic worst-case perspective.

The computation of efficient schedules in the SINR model has been studied in previous work in various flavors. The work of [36] proposes a mathematical programming formulation for deriving optimal schedules. However,

the resulting formulations are infeasible from a computational point of view. The authors then propose a heuristic based on a so-called column generation approach, which they show to produce fast schedules in practical scenarios. Finally, it is shown in [36] that the problem of deriving optimal schedules is NP-hard, even in a much more restricted model. The works of [34, 35, 131] also derive mathematical programming formulations and investigates the impact of power assignments to nodes on the achievable throughput capacity.

In [81, 112, 113], various protocols for scheduling in SINR-based models have been proposed and evaluated under different traffic and random node distribution models. These protocols being evaluated by means of *simulation*, none of them provides theoretical bounds on the competitiveness in a worst-case sense. The algorithms in [62, 73] study the problem of finding schedule and power control policies that minimize the total average transmission power in the wireless multi-hop network. The algorithm in [73], for instance, is based on guaranteeing a certain "spatial reuse" distance between all pairs of simultaneously transmitting nodes. As we will see in Chapter 18, such an approach inherently cannot yield competitive results in worst-case networks. For an exploration of spatial reuse TDMA in SINR-based models, we refer to the thesis by Grönkvist [113]. That is, all currently known protocols may produce schedules such that, in certain networks and for certain request sequences, can be significantly worse than the optimal solution.

In summary, there has so far been no work that considers scheduling protocols in SINR-based models from an *algorithmic* point of view, i.e., with provable guarantees on their competitiveness and the actual scheduling complexity of natural properties and topologies in wireless networks has been completely unresolved. All existing protocols do either not yield provable worst-case guarantees or are based on solutions to complex optimization problems that are computationally intractable even for small networks.

In the subsequent chapters, we are interested in obtaining scaling laws that describe the asymptotic behavior of the scheduling complexity as the network grows. We also seek to devise scheduling algorithms that achieve good performance with regard to the scheduling complexity since such algorithms would come close to being an "optimal" or at least competitive MAC layer protocol. As it turns out, studying the scheduling complexity of different protocols reveals previously unknown and practically important aspects of communication in wireless networks.

Chapter 18

Inefficiency of Simple MAC Layer Protocols

The task faced by any MAC layer or scheduling protocol is twofold. The protocol not only decides which nodes transmit in which time slot, it also assigns proper power levels. As it turns out, particularly the second task—assigning transmission powers—is non-trivial. In this section, we prove a striking deficiency of all protocols and power assignment schemes that have been widely studied in the field of wireless networks (and that have also been adopted by all standard MAC layer protocols). Specifically, the scheduling complexity achieved by such protocols can be $\Theta(n)$ times worse than the optimum, even for simple request sequences.

In order to derive this result, we ask the following seemingly simple question: *How much time is required until every node can successfully transmit one message, when the receivers for each sender are selected best possible?* Or in other words, how much time is required until every node can communicate its identifier or even one bit of information to some other node in the network? Technically speaking, we want to explore the scheduling complexity of the following simple scheduling task Ψ_{min}: *Every node $v \in V$ can send at least one message successfully.* Note again that Ψ_{min} does not restrict to *which* other node a node must send, i.e., every nodes can for instance select its nearest neighbors as receiver. That is, all we want to know is the maximum number of time slots required in a wireless communication medium until every node can send something to someone.

Because of its extreme simplicity, understanding the scheduling complexity of this scheduling task Ψ_{min} is fundamental: If a MAC protocol is not efficient (competitive) for this simple task, it cannot possibly be efficient for any realistic scheduling task where there may be traffic flows or routing requirements. Moreover, achieving good solutions for Ψ_{min} (i.e., an algorithm with low scheduling complexity for Ψ_{min}) appears to be easy and intuitively, one would expect standard MAC layer protocols to achieve an excellent per-

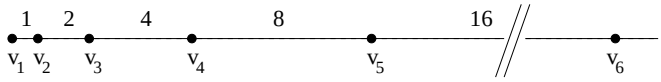

Figure 18.1: Example with nodes v_i being located at position $v_i = 2^i$, $i = 1, \ldots, n$.

formance for this problem.

Surprisingly, however, the opposite is true: All generally employed and studied power assignment schemes are incapable of achieving a reasonable scheduling complexity even for the simple scheduling task Ψ_{min}. In the worst case, these intuitive protocols can essentially be as slow as the trivial protocol that schedules every single node individually, without taking advantage of spatial reuse at all. In the sequel, we consider the two most common power assignment policies: uniform and linear power assignment.

18.1 Uniform Power Assignment

Possibly the simplest way of assigning power levels in a radio network is to let every node transmit at the same power. Such *uniform power assignment* schemes have been widely studied (e.g. [116, 214]) and adopted in practical systems. Moreover, standard graph-based models such as the *unit disk graph* implicitly assume uniform power assignment. The following theorem states, however, that even for the simple scheduling task Ψ_{min}, the scheduling complexity of any uniform power assignment algorithm is *linear in n*.

Theorem 18.1. *Assume that every node $v_i \in V$ has the same transmission power. The scheduling complexity of Ψ_{min} achieved by any protocol using uniform power assignment is at least $n \cdot \frac{\beta}{2^\alpha + \beta} \in \Omega(n)$, even in the absence of ambient noise.*

Proof. Consider the example network given in Figure 18.1, in which nodes $v_0, \ldots, v_{n-1}$ are placed on a straight line with exponentially increasing distances between them. We prove that in each time slot, at most $\frac{2^\alpha}{\beta} + 1$ nodes can send successfully if the transmission power is uniform. Assume for contradiction that there are $L = \frac{2^\alpha}{\beta} + 2$ nodes sending successfully in the same time slot, and let v_s be the right-most of these transmitters. Further, assume that v_s's transmission is successfully received by node v_r. On an exponential line, if v_r is to the left of v_s, it holds that $d(v_i, v_r) \leq d(v_s, v_r)$ for each simultaneously transmitting node v_i. If v_r is on v_s's right, it holds that $d(v_i, v_r) \leq 2d(v_s, v_r)$ for each such v_i. Because all transmission powers P are equal and v_s is the right-most sender, the SINR at v_r is hence at most

$$\frac{\frac{P}{d(v_s,v_r)^\alpha}}{N + (L-1) \cdot \frac{P}{(2d(v_s,v_r))^\alpha}} \leq \frac{2^\alpha}{L-1} = \frac{2^\alpha \beta}{2^\alpha + \beta} < \beta,$$

which is not sufficiently high for a correct reception of the message at v_r, which yields the contradiction. Because at most $\frac{2^\alpha}{\beta}+1$ links can be simultaneously scheduled in any time slot, the algorithm requires at least $n\cdot(\frac{2^\alpha+\beta}{\beta})^{-1}$ time slots to schedule all nodes at least once, from which the theorem follows. □

18.2 Linear $\mathbf{P \sim d^\alpha}$ Power Assignment

The other intuitive and frequently adopted way of assigning power levels when scheduling a set of wireless nodes is the following: Intended senders transmit at a power level that is *proportional* to the minimal power *required* for transmitting over the wireless link (e.g., [27, 178, 228, 234]). In other words, for a pair of sender s_i and receiver r_i, s_i sends with power $P_s = \rho\cdot d(s_i,r_i)^\alpha$, where ρ is an arbitrary constant which may depend on the values of α, β, and the ambient noise N. Since $d(s_i,r_i)^\alpha$ is the minimal amount of power necessary to reach r_i from s_i, it seems natural to let nodes send with a power that is proportional to $d(s_i,r_i)^\alpha$, in order to avoid unnecessary interference. We call such a power assignment *linear*, because the power assigned to a node depends linearly on the minimal power required for its link. Like in the uniform case, however, any protocol using linear power assignment can perform badly even for the basic task Ψ_{min}.

Linear power assignments have been assumed in many papers written on topology control (e.g. [164, 204]), in papers proposing energy efficient protocols for wireless networks (e.g. [27, 228, 234]), and in some MAC layer protocols [178]. Moreover, linear power assignments have often implicitly been assumed in theoretical studies regarding energy efficient network design, for instance in the so-called minimum energy broadcast problem, e.g., [12, 56, 57].

Theorem 18.2. *Assume that every node v_i that intends to send a message over a link of length ℓ_i transmits with power $P_s = \rho\cdot\ell_i^\alpha$, for an arbitrary constant ρ which may depend on α, β, or N. The scheduling complexity for Ψ_{min} achieved by any protocol using such a linear power assignment is $n\cdot\min\{1,\beta/2^\alpha\}\in\Omega(n)$, even in the absence of ambient noise.*

Proof. Consider again the example given in Figure 18.1. Let v_i be a transmitting node in an arbitrary time slot t. In a linear power assignment, it transmits with power $P_i = \rho\cdot d(v_i,v_{i-1})^\alpha$, for some constant ρ. Consequently, all nodes $v_j, j<i$ face an interference of at least

$$I_j(v_i) \;\geq\; \frac{\rho\cdot d(v_i,v_{i-1})^\alpha}{(2d(v_i,v_{i-1}))^\alpha} \;=\; \frac{\rho}{2^\alpha}$$

because the distance $d(v_i,v_j)$ is at most $2d(v_i,v_{i-1})$ in the exponential line. Because at least the same amount of interference is caused by all simultaneous senders v_i, a node v_j faces a total interference of at least $I_j \geq R\cdot\frac{\rho}{2^\alpha}$, where R is the number of sending nodes to the right of v_j. Now, let v_s be the

left-most node that sends a message in time slot t, and let v_r be its receiver. Because the SINR at v_r must be at least β, it must hold that

$$\frac{\frac{\rho \cdot d(v_s,v_r)^\alpha}{d(v_s,v_r)^\alpha}}{N + R \cdot \frac{\rho}{2^\alpha}} \geq \frac{\rho 2^\alpha}{2^\alpha N + \rho R} \stackrel{!}{\geq} \beta.$$

From this, it follows that the maximum number of simultaneous senders R_{max} can be at most $R_{max} \leq \frac{2^\alpha}{\beta}$ and consequently, the algorithm requires at least $n \cdot \min\{1, \beta/2^\alpha\}$ time slots for scheduling all nodes. Note that this result holds even if there is no noise N. □

Discussion

In reality, both α and β are small constant values and therefore, Theorems 18.1 and 18.2 show that even in the most basic scheduling problem Ψ_{min}, only a *small constant number* of links can be simultaneously scheduled when adopting uniform or linear power assignment schemes. For $\alpha = 4$ and $\beta = 7dB$, for instance, at most 4 links can be scheduled in parallel. On the other hand, it can be shown that there exist schedules of *constant length* for successfully scheduling *all links* in the topology of Figure 18.1. It follows that any MAC layer or scheduling protocol that assigns transmission powers according to either of these two policies may perform by a factor $\Theta(n)$ times worse than an optimal protocol. Specifically, the subsequent Chapter 19 will show how the use of a highly *non-linear power assignment* yields much more efficient, and provably competitive schedules. Like in the example discussed in Section 17.2, the idea is to "over-power" nodes with small links in order to guarantee a high degree of parallelism and spatial reuse.

The bad scheduling complexity achieved by uniform and linear power assignments has practical relevance: It shows that in order to obtain a competitively fast scheduling of sending requests in wireless networks, MAC layer and scheduling protocols must adopt neither uniform nor linear power assignment. Instead, the remedy against this loss of efficiency is a less intuitive, highly non-linear assignment of power levels; a power scheme that lies "in between" uniform and linear power assignment. Particularly, *many different power levels* are required in order to achieve efficient schedules.

The lower bounds also challenge a common assumption made in the theoretical literature on wireless ad hoc and sensor networks. Specifically, it is often argued that the energy required for transmitting a message over a distance d is in the order of d^α. Based on this fact, problems such as the *minimum energy broadcast* have been studied using a so-called *energy-metric* in which the cost of each link of length d corresponds to d^α [12, 56, 57, 164, 228]. While this analytical approach certainly leads to interesting concepts and insights, one has to be aware that if nodes really transmitted using the power-levels implied by the energy-metric, the resulting scheduling would be highly sub-optimal. In this sense, abstracting away the aspect of scheduling when studying *energy-metrics* in a static, graph-theoretical way is problematic.

Chapter 19

Polylogarithmic Scheduling Complexity

In this section, we show that when using proper power assignment policies, much more efficient scheduling is possible even in worst-case networks. Specifically, we present a protocol that achieves a scheduling complexity of $O(\log^2 n)$ for the simple scheduling task Ψ_{min} in arbitrary networks, thus exponentially improving on all protocols that employ uniform or linear power assignment schemes. In fact, we show that even much more complex scheduling tasks can be scheduled in polylogarithmic time in wireless networks. In particular, we study the *scheduling complexity of connectivity* in wireless networks. That is, we seek to determine the minimum number of time slots required for successfully scheduling a strongly-connected topology. Section 19.1 shows that even in worst case networks, a strongly-connected topology can be successfully scheduled in at most $O(\log^3 n)$ time slots.

The lower bounds in Chapter 18 imply that algorithms based on linear and uniform power assignment schemes inherently have a scheduling complexity of at least $\Omega(n)$. On the other hand, MAC layer protocols based on standard power assignment strategies have the practical advantage that their implementation is simple. We therefore examine a simple scheduling algorithm that employs linear power assignments in Section 19.2.

19.1 The Complexity of Connectivity

In the previous chapter, we have seen that neither uniform nor linear power assignment schemes achieve a polylogarithmic scheduling complexity. Therefore, in order to obtain the claimed result, our algorithm employs a power-assignment policy that favors short links over long links in the sense that senders with short links transmit at a power level that is significantly higher than required in the absence of interference.

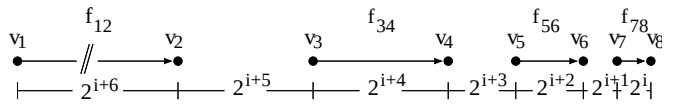

Figure 19.1: An example in which scheduling many requests in parallel requires a non-trivial power assignment policy.

Let the Euclidean diameter $\mathcal{D}$ be the largest distance between any two nodes in the network and let $\ell(f_{ij})$ denote the length of a link f_{ij} from node v_i to v_j.

In order to give a first intuitive insight into the power assignment used by Algorithm 19.1, consider the instance depicted in Figure 19.1. For the following considerations we momentarily neglect the presence of ambient noise N for simplicity. If any of the four links are to be scheduled individually, setting the transmission power of the respective sender v_i to $\beta \cdot \ell(f_{i,i+1})^\alpha$ is sufficient for the signal to be correctly received in the absence of interference. Should more than one link be scheduled simultaneously, however, the situation becomes more intricate. If link f_{12} is scheduled successfully, the signal power received by v_2 is at least β, and hence the intended receivers of all other links face an interference of at least $\beta/2^\alpha$. It follows that if we want some of the smaller links to be scheduled simultaneously, every sender v_i of any one of these links must transmit at a power that is at least by a factor $\beta/2^\alpha$ greater than $\beta \cdot \ell(f_{i,i+1})^\alpha$. The problem is that the interference created by long link cascades, that is, if all four links are scheduled in the same time slot, the third and fourth senders (v_5 and v_7) must transmit with a power of at least $\beta^3/2^{2\alpha} \cdot \ell(f_{5,6})^\alpha$ and $\beta^4/2^{3\alpha} \cdot \ell(f_{7,8})^\alpha$, respectively, in order to guarantee successful simultaneous reception.[1] This observation weighs particularly heavily for the following reason: If we want fast, say, polylogarithmic schedules, there must exist time slots in which at least $n/\log^c n$ nodes are scheduled simultaneously for some constant c. The dependence of the chosen transmission power on other simultaneously scheduled links—together with the necessity to schedule relatively many links at the same time—shows that every provably efficient scheduling protocol must inevitably employ a complex and sophisticated power assignment strategy.

In Algorithm 19.1, the transmission power of a node v_i transmitting over a link f_{ij} is scaled by a factor of $(3n\beta)^{\tau(f_{ij})}$, where $\tau(f_{ij})$ is a value that reflects the relative position of $\ell(f_{ij})$ in an ordering of all link lengths. Unfortunately, scaling up the transmission powers of nodes with short links in turn entails new problems. Since a node v_i with a short link f_{ij} now transmits at a power that is high relative to $\ell(f_{ij})$, v_i may cause significant interference at a receiver v_h even if the distance $d(v_i, v_h)$ is exponentially larger than $\ell(f_{ij})$. In other words, the unavoidable scaling of transmission powers renders simple

[1] For our illustration we assume that $\beta > 2^\alpha$; otherwise the node distances can be adapted to produce a similar situation in which the nodes' transmission powers are disproportionate compared to their radii.

geometric arguments based on reuse distances problematic.

With regard to our studies of locality in Part I of this thesis, it is interesting to observe that this *non-locality of interference* in the SINR model is in stark contrast to all generally studied graph-based interference models. In fact, the above intuition shows that *fast scheduling in the SINR model* is inherently a *non-local* task and simple local approaches that typically work in graph-based models fail to produce reasonable solutions.

The Algorithm

In every, possibly worst-case network, Algorithm 19.1 computes a schedule of length at most $O(\log^3 n)$ time slots in which all scheduled links combine for a strongly-connected topology. The algorithm proceeds in *phases*, each phase corresponding to an iteration of the outermost loop. The purpose of this outer loop is to gradually reduce the number of *active nodes* $v_i \in \mathcal{A}$. Initially, the set of active nodes $\mathcal{A}$ contains all nodes, and whenever a node becomes passive (by being discarded from $\mathcal{A}$), it does not transmit in any subsequent time slot. At the outset of a phase p, every active node v_i chooses its *closest* active neighbor, say v_j, and the directed link $f_{ij} = (v_i, v_j)$ becomes designated to be scheduled in phase p (Lines 6 and 7). After breaking cycles of length 2 (i.e. two nodes that are mutually closest neighbors) in Line 7, $\mathcal{F}_p$ is the set of all selected links that are to be scheduled in phase p. $\mathcal{F}_p$ forms a *nearest neighborhood forest* consisting of a set of trees, from each of which only the root remains active in the next phase $p+1$. This process is repeated until there remains only a single active node. At this point, the scheduled links form a *directed tree* towards a single node, which can then complete the strong connectivity requirement in a single additional time slot.

The main challenge, however, is how to quickly schedule the forest $\mathcal{F}_p$. In the sequel, we describe the algorithm for efficiently scheduling $\mathcal{F}_p$ on a more technical level. At the outset of the **schedule**() procedure of Algorithm 19.1, the algorithm partitions the set of links $\mathcal{F}_p$ into at most $\lfloor \log \mathcal{D} \rfloor + 1$ possibly empty disjoint sets $\mathcal{L} = L_0, \ldots, L_{\lfloor \log \mathcal{D} \rfloor}$. Each such set L_h contains every link $f_{ij} \in \mathcal{F}_p$ with length $2^h \leq \ell(f_{ij}) < 2^{h+1}$. If no such link exists, the set L_h remains empty. In the next step, the algorithm removes all these empty sets and *renames* the remaining *non-empty sets* such that L_h is the h^{th} non-empty set in decreasing order of the lengths of the contained links, for $h = 1, 2, \ldots$ (see Figure 19.2). In the resulting partition, the lengths of all links in the same set are still within a factor of 2, whereas the length of two links in sets L_h and L_{h+1} may differ by an arbitrarily large factor if many empty sets were deleted between L_h and L_{h+1}.

The task of each of the $\lceil \log(3n\beta) \rceil$ iterations of the subsequent for-loop is to schedule a subset of all the links. In particular, in the k^{th} iteration of the loop, links in sets $L_{m\lceil \log(3n\beta) \rceil + k}$ are scheduled for all integers m. All these links form the set of links $\mathcal{F}_k$ that is to be scheduled in the k^{th} iteration. The reason for partitioning the entire set of links into $\lceil \log(3n\beta) \rceil$ subsets is to guarantee (cf. Lemma 19.6) that two links scheduled in the same time slot either have almost the same length (when they are in the same set of the

1: $\mathcal{A} := V$; $t := 1$; Define constants $\nu := 4N$ and $\mu := 3 + 2^{\frac{7}{\alpha}+2} \sqrt[\alpha]{\frac{\beta(\alpha-1)}{\alpha-2}}$;
2: **while** $|\mathcal{A}| > 1$ **do** {* *Phase p* *}
3: $\mathcal{F}_p := \emptyset$;
4: **for each** $v_i \in \mathcal{A}$ **do**
5: choose $v_j \in \mathcal{A} \setminus \{v_i\}$ minimizing $d(v_i, v_j)$;
6: $f_{ij} := (v_i, v_j)$; $\ell(f_{ij}) := d(v_i, v_j)$;
7: **if** $f_{ji} \notin \mathcal{F}_p$ **then** $\mathcal{F}_p := \mathcal{F}_p \cup f_{ij}$; **fi**
8: **end for**
9: **for each** $v_i \in \mathcal{A}$ with $f_{ij} \in \mathcal{F}_p$ **do** $\mathcal{A} := \mathcal{A} \setminus \{v_i\}$;
10: **schedule**($\mathcal{F}_p$);
11: **end while**
12: $\phi_t(v_i) := N\beta \cdot \mathcal{D}^\alpha$ for $v_i \in \mathcal{A}$
13: $\mathcal{S} := \{\phi_1, \ldots, \phi_{t-1}\}$;

schedule($\mathcal{F}_\mathbf{p}$)
1: Partition $\mathcal{F}_p$ into sets $\mathcal{L} = L_0, \ldots, L_{\lfloor \log \mathcal{D} \rfloor}$ such that L_h contains all links f_{ij} with $2^h \leq \ell(f_{ij}) < 2^{h+1}$;
2: Delete all empty sets $L_h \in \mathcal{L}$ and rename $\mathcal{L}$ such that L_h is the h^{th} non-empty set in *decreasing* order of the lengths of the contained links;
3: **for** $k = 1$ **to** $\lceil \log(3n\beta) \rceil$ **do**
4: Let $\mathcal{F}_k$ be the union of all sets $L_{m\lceil \log(3n\beta) \rceil + k} \in \mathcal{L}$ for $m \in \mathbb{N}_0$;
5: **for each** $f_{ij} \in \mathcal{F}_k$ **do**
6: $\tau(f_{ij}) := \chi$, where $f_{ij} \in L_\ell$ and L_ℓ is the χ^{th} set in $\mathcal{F}_k$ (in decreasing order of lengths);
7: **end for**
8: **while** not all links in $\mathcal{F}_k$ have been scheduled **do**
9: $E_t := \emptyset$;
10: Consider all links $f_{ij} \in \mathcal{F}_k$ in decreasing order of $\ell(f_{ij})$:
11: **if** *allowed*(f_{ij}, E_t) **then** $E_t := E_t \cup \{f_{ij}\}$;
12: Schedule all $f_{ij} \in E_t$ in time slot t and assign v_i a transmission power of $P_i = \phi_t(v_i) := \nu(3n\beta)^{\tau(f_{ij})} \cdot \ell(f_{ij})^\alpha$;
13: Remove all scheduled links ($\mathcal{F}_k := \mathcal{F}_k \setminus E_t$);
14: $t := t + 1$;
15: **end while**
16: **end for**

allowed($\mathbf{f_{ij}}, \mathbf{E_t}$)
1: **for each** $f_{ab} \in E_t$ **do**
2: $\delta_{ia} := \tau(f_{ij}) - \tau(f_{ab})$;
3: **if** $\tau(f_{ij}) = \tau(f_{ab})$ and $\mu \cdot \ell(f_{ij}) > d(v_i, v_a)$ **return false**
4: **else if** $\ell(f_{ij}) \cdot (3n\beta)^{(\delta_{ia}+1)/\alpha} > d(v_i, v_b)$ **return false**
5: **end for**
6: **return true**

Algorithm 19.1: Polylogarithmic Scheduling Algorithm

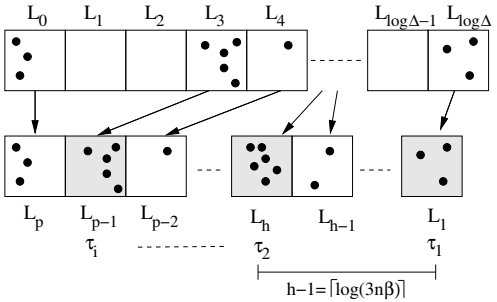

Figure 19.2: Naming of the sets in partition $\mathcal{L}$. The uppermost row represents the names given in Line 1 of the **schedule**() procedure, the second row reflects the situation after the renaming in Line 2, and the τ_i at the bottom denotes the power scaling factors assigned in Line 6.

partition $\mathcal{L}$) or their lengths differ significantly. We will use this property in the key Lemma 19.7.

It may not be possible to schedule all links in $\mathcal{F}_k$ in a single time slot. Even scheduling the links of $\mathcal{F}_k$ alone turns out to be a challenging task, as we show in the following. In Lines 5 and 6 of the **schedule**() procedure, each link $f_{ij} \in \mathcal{F}_k$ designated to be scheduled in this iteration of the for-loop determines its $\tau(f_{ij})$ value. If link f_{ij} is in the set with the largest lengths of the sets selected in $\mathcal{F}_k$, $\tau(f_{ij})$ is set to 1. Or generally speaking, if a link f_{ij} is in set L_ℓ and L_ℓ is the χ^{th} set (still in decreasing order of lengths) of all sets forming $\mathcal{F}_k$, then $\tau(f_{ij}) := \chi$ (cf. Figure 19.2). Intuitively, short links have a large $\tau(f_{ij})$, while long links have a small $\tau(f_{ij})$. In other words, $\tau(f_{ij})$ is a power scaling factor reflecting the fact that—as illustrated in the example of Figure 19.1—nodes with short links may have to send with disproportionately high transmission powers compared to nodes with long links.

At the heart of Algorithm 19.1 is the subsequent while-loop which schedules all nodes in $\mathcal{F}_k$ using essentially at most $O(\log n)$ time slots, as shown later in Lemmas 19.5 and 19.7. The set of links scheduled in parallel in time slot t is denoted by E_t; at the end of each iteration, all links that were scheduled are removed from $\mathcal{F}_k$ (Line 13). The selection of links for E_t proceeds as follows. Links in $\mathcal{F}_k$ are considered one by one in decreasing order of $\ell(f_{ij})$. When considering a link f_{ij}, the algorithm checks whether scheduling f_{ij} conflicts with previously selected (longer) links in E_t in the sub-procedure **allowed**($\mathbf{f_{ij}}, \mathbf{E_t}$). This procedure returns true if and only if

1. Link f_{ij} can be successfully scheduled in spite of the interference created by links already in E_t.

2. All senders in E_t can still successfully transmit in spite of the additional interference caused by f_{ij}.

As shown in the analysis, these two properties can be guaranteed by requiring that for all $f_{ab} \in E_t$, it holds that $d(v_i, v_a) \geq \ell(f_{ij}) \cdot \mu$ if $\tau(f_{ij}) = \tau(f_{ab})$ or $d(v_i, v_b) \geq \ell(f_{ij}) \cdot (3n\beta)^{(\delta_{ia}+1)/\alpha}$ otherwise, where $\delta_{ia} = \tau(f_{ij}) - \tau(f_{ab})$ and the constant μ is set to a large enough value as to ensure low interference.

Clearly, Algorithm 19.1 is not primarily intended for being employed as a practical network protocol in its current form. Besides being rather complex, the necessity that each node v_i knows its corresponding (non-local) value $\tau(f_{ij})$ for the purpose of determining its own transmission power renders the algorithm non-trivial to implement in a distributed way. However, the algorithm shows that, theoretically, complex requests can be scheduled efficiently also in large-scale worst-case networks.

Analysis

The analysis of Algorithm 19.1 consists of two parts. First, we need to guarantee that the obtained schedule $\mathcal{S}$ is valid, that is, all links are successfully received by the intended receivers. Second, we prove that the number of time slots required in the worst case does not exceed $O(\log^3 n)$.

In order to guarantee that every link f_{ij} scheduled by the algorithm in a time slot t_s can be correctly received by the intended receiver v_j, we bound the total interference accrued at the receiver. For this purpose, we first bound the interference created by simultaneously scheduled links f_{ab} that are significantly longer.

Lemma 19.1. *Consider a time slot t_s in which the algorithm schedules a link f_{ij} for transmission. It holds for the intended receiver v_j and for any simultaneously scheduled link $f_{ab} \in \mathcal{F}_k \setminus \{f_{ij}\}$ with $\tau(f_{ab}) < \tau(f_{ij})$ that*

$$I_j(v_a) \leq \nu(3n\beta)^{\tau(f_{ij})-1}.$$

Proof. In $\mathcal{F}_k$, every node has a link to its closest neighbor and hence, $\ell(f_{ab}) \leq d(v_j, v_a)$ for all links f_{ab}. The interference at v_j caused by v_a is therefore at most

$$\begin{aligned} I_j(v_a) &= \frac{P_a}{d(v_j, v_a)^\alpha} \leq \frac{\nu(3\beta n)^{\tau(f_{ab})}\ell(f_{ab})^\alpha}{\ell(f_{ab})^\alpha} \\ &= \nu(3\beta n)^{\tau(f_{ab})} \leq \nu(3\beta n)^{\tau(f_{ij})-1}. \end{aligned}$$

□

In the following lemma we show that the interference caused by concurrently scheduled links that are significantly *shorter* than f_{ij} is equally bounded.

Lemma 19.2. *Consider a time slot t_s in which the algorithm schedules a link f_{ij} for transmission. It holds for the intended receiver v_j and for any simultaneously scheduled link $f_{ab} \in \mathcal{F}_k \setminus \{f_{ij}\}$ with $\tau(f_{ab}) > \tau(f_{ij})$ that*

$$I_j(v_a) \leq \nu(3n\beta)^{\tau(f_{ij})-1}.$$

Proof. The interference $I_j(v_a)$ incurred by a node v_a at v_j is at most

$$I_j(v_a) \leq \frac{\nu(3n\beta)^{\tau(f_{ab})} \cdot \ell(f_{ab})^\alpha}{d(v_a, v_j)^\alpha}.$$

Assume for contradiction that there exists a link f_{ab} with $\tau(f_{ab}) > \tau(f_{ij})$ and $I_j(v_a) > \nu(3n\beta)^{\tau(f_{ij})-1}$. Then

$$\frac{\nu(3n\beta)^{\tau(f_{ab})} \cdot \ell(f_{ab})^\alpha}{d(v_a, v_j)^\alpha} > \nu(3n\beta)^{\tau(f_{ij})-1}$$

holds and hence $d(v_a, v_j) < \sqrt[\alpha]{(3n\beta)^{\tau(f_{ab})-\tau(f_{ij})+1}} \cdot \ell(f_{ab})$. Because it holds by the triangle inequality that $d(v_a, v_i) \leq d(v_a, v_j) + \ell(f_{ij})$,

$$\begin{aligned} d(v_a, v_i) &< \sqrt[\alpha]{(3n\beta)^{\tau(f_{ab})-\tau(f_{ij})+1}} \cdot \ell(f_{ab}) + \ell(f_{ij}) \\ &= \ell(f_{ab}) \cdot (3n\beta)^{\frac{\delta_{ai}+1}{\alpha}} + \ell(f_{ij}) \end{aligned}$$

follows. However, this contradicts the fact that v_a and v_i are selected for scheduling in the same time slot. Particularly, at the time $\mathbf{allowed}(\mathbf{f_{ab}}, \mathbf{E_t})$ is invoked, f_{ij} is already in E_t, and the procedure would evaluate to *false*. Hence, f_{ab} and f_{ij} cannot be scheduled in the same time slot. □

With these two lemmas, we can now establish that throughout the algorithm, every scheduled link is successfully received by the intended receiver.

Lemma 19.3. *For every k and for every link $f_{ij} \in \mathcal{F}_k$, there exists a unique time slot t_s in which v_j successfully receives a message from v_i.*

Proof. We begin by showing that every f_{ij} is scheduled exactly once during the execution of Algorithm 19.1. Every f_{ij} belongs to a single set L_h and each such set is considered in exactly one iteration of the outermost for-loop (more precisely, set L_h is scheduled in iteration k in which $h = m \log(3\beta n) + k$ for some integer $m \geq 0$). Consider this iteration: As long as f_{ij} is not scheduled, it remains in $\mathcal{F}_k$ and the while-loop (Lines 8–15) does not terminate. Termination of this while-loop is guaranteed, however, by the fact that in every iteration at least the *longest link* in $\mathcal{F}_k$ is selected for scheduling in E_t and consequently removed from $\mathcal{F}_k$. Because after at most n iterations the set $\mathcal{F}_k$ is empty and the loop terminates, there must be a time slot in which f_{ij} is scheduled. Further, note that since every $f_{ij} \in E_t$ is removed from $\mathcal{F}_k$, every link has a unique time slot t_s in which it is scheduled.

Hence, we need to prove that in this time slot t_s, the message is received successfully by the intended receiver v_j. For this purpose, we bound the total interference $I_j = \sum_{v_a \in V \setminus \{v_i\}} I_j(v_a)$ experienced by any such receiver.

By Lemmas 19.1 and 19.2 we know that for all f_{ab} with $\tau(f_{ab}) < \tau(f_{ij})$ and $\tau(f_{ab}) > \tau(f_{ij})$, the interference $I_j(v_a)$ is bounded by $\nu(3n\beta)^{\tau(f_{ij})-1}$.

Hence, because there are at most n nodes in these sets, it holds that

$$\sum_{v_a:\tau(f_{ij})\neq\tau(f_{ab})} I_j(v_a) \quad \leq \quad n \cdot \nu(3n\beta)^{\tau(f_{ij})-1}. \tag{19.1}$$

What remains to be bound is the interference created by concurrently scheduled links f_{ab} for which $\tau(f_{ab}) = \tau(f_{ij})$, that is by nodes that are in the same length class of the partition $\mathcal{L}$.

Let $\mathcal{T}$ be the set of simultaneously scheduled links f_{ab} with $\tau(f_{ij}) = \tau(f_{ab})$. Abusing notation, we also write $v_a \in \mathcal{T}$ to denote that node v_a is an intended sender with link in $\mathcal{T}$. By the definition of $\mathbf{allowed}(\mathbf{f_{ab}}, \mathbf{E_t})$ a link $f_{ab} \in \mathcal{T}$ prevents all links $f_{cd} \in \mathcal{T}$ for which $\mu \cdot \ell(f_{cd}) > d(v_a, v_c)$ from being added to $\mathcal{T}$. Because the lengths of all links in $\mathcal{T}$, including f_{ij}, differ at most by a factor of 2, it follows that around each $f_{ab} \in \mathcal{T}$ there can be no other scheduled sender v_c in $\mathcal{T}$ within distance less than $\frac{1}{2}\mu \cdot \ell(f_{ab})$. Hence, disks D_i of radius $\frac{1}{4}\mu\ell(f_{ab})$ centered at every sender $v_a \in \mathcal{T}$ do not overlap.

Consider rings R_λ of width $\mu\ell(f_{ij})$ around v_i, that is, R_λ contains all intended transmitters $v_a \in \mathcal{T}$ for which $(\lambda - \frac{1}{2})\mu\ell(f_{ij}) < d(v_i, v_a) \leq (\lambda + \frac{1}{2})\mu\ell(f_{ij})$. Consider all transmitters $v_a \in \mathcal{T} \cap R_\lambda$ for some integer $\lambda > 0$. All corresponding disks D_i must be located entirely in an extended ring of area

$$\begin{aligned} A(R_\lambda^+) &= \left[\left(\left(\lambda + \frac{3}{4}\right)\mu\ell(f_{ij})\right)^2 - \left(\left(\lambda - \frac{3}{4}\right)\mu\ell(f_{ij})\right)^2\right]\pi \\ &= 3\lambda\mu^2\ell(f_{ij})^2\pi. \end{aligned}$$

The distance of a transmitter in R_λ to v_j is at least $((\lambda - \frac{1}{2})\mu - 1)\ell(f_{ij})$, and each such node transmits with a power of at most $\nu(3n\beta)^{\tau(f_{ij})} \cdot (2\ell(f_{ij}))^\alpha$. By applying a standard geometric area argument, we can bound the total interference $I_\lambda = \sum_{v_a \in \mathcal{T}\cap R_\lambda} I_j(v_a)$ incurred by nodes $v_a \in \mathcal{T} \cap R_\lambda$ as

$$\begin{aligned} I_\lambda &\leq \frac{A(R_\lambda^+)}{A(D_i)} \cdot \frac{\nu(3n\beta)^{\tau(f_{ij})} \cdot (2\ell(f_{ij}))^\alpha}{(((\lambda - \frac{1}{2})\mu - 1)\ell(f_{ij}))^\alpha} \\ &< \frac{3\lambda\mu^2\ell(f_{ij})^2\pi}{(\frac{1}{4}\mu\ell(f_{ij}))^2\pi} \cdot \frac{\nu(3n\beta)^{\tau(f_{ij})} \cdot (2\ell(f_{ij}))^\alpha}{(\frac{1}{2}\lambda(\mu-1)\ell(f_{ij}))^\alpha} \\ &= \frac{48\nu(3n\beta)^{\tau(f_{ij})}2^{2\alpha}}{\lambda^{\alpha-1}(\mu-1)^\alpha}. \end{aligned}$$

Summing up the interference over all rings R_λ, we obtain

$$\begin{aligned} \sum_{\lambda=1}^{\infty} I_\lambda &\leq \frac{48\nu(3n\beta)^{\tau(f_{ij})}2^{2\alpha}}{(\mu-1)^\alpha} \sum_{\lambda=1}^{\infty} \frac{1}{\lambda^{\alpha-1}} \\ &< \frac{48\nu(3n\beta)^{\tau(f_{ij})}2^{2\alpha}}{(\mu-1)^\alpha} \cdot \frac{\alpha-1}{\alpha-2} \\ &< \nu \cdot (3\beta)^{\tau(f_{ij})-1} \cdot n^{\tau(f_{ij})}, \end{aligned}$$

where the second-to-last inequality follows from a standard bound for Riemann's zeta-function and $\alpha > 2$. The last inequality is derived by plugging in the definition of μ.

Adding up the total interference created by senders v_a whose links satisfy $\tau(f_{ab}) = \tau(f_{ij})$ and the total interference by all other nodes as bounded in Inequality (19.1), we obtain

$$\begin{aligned} I_r &\leq \nu(3\beta)^{\tau(f_{ij})-1} \cdot n^{\tau(f_{ij})} + \nu(3\beta)^{\tau(f_{ij})-1} \cdot n^{\tau(f_{ij})} \\ &= \frac{2\nu}{3} \cdot \beta^{\tau(f_{ij})-1} \cdot (3n)^{\tau(f_{ij})}. \end{aligned}$$

Finally, the SINR experienced at any intended receiver v_j of a link f_{ij} is therefore at least

$$\begin{aligned} SINR_j &\geq \frac{\nu(3n\beta)^{\tau(f_{ij})}}{N + \frac{2}{3}\nu\beta^{\tau(f_{ij})-1} \cdot (3n)^{\tau(f_{ij})}} \\ &= \frac{4(3n\beta)^{\tau(f_{ij})}}{1 + \frac{8}{3}\beta^{\tau(f_{ij})-1} \cdot (3n)^{\tau(f_{ij})}} \\ &\geq \frac{4(3n\beta)^{\tau(f_{ij})}}{\frac{11}{3}\beta^{\tau(f_{ij})-1} \cdot (3n)^{\tau(f_{ij})}} > \beta, \end{aligned}$$

where the second inequality follows from the definition of ν and the third inequality from the fact that $n, \beta \geq 1$. Hence, all scheduled messages are received correctly. □

We now turn our attention to the second aspect of the analysis. Particularly, we prove that the number of time slots required by Algorithm 19.1 is small, and hence, the scheduling complexity is low. We require the following geometric lemma.

Lemma 19.4. *Consider a disk C with radius r_c, and disks D_i with centers s_i and radius r_i, $r_i \geq r_c$ for all i. Let κ be the maximal number of such disks D_i such that both of the following properties hold:*

- *Every D_i overlaps with C in at least one point.*
- *No disk D_i contains a center c_j for $i \neq j$.*

Then, it holds that $\kappa \leq 36$.

Proof. The proof follows a geometry argument. Assume for contradiction that there are $\kappa + 1$ disks D_i that fulfill the properties stated in the lemma and consider the corresponding centers c_i. There must be a cone of angle $\frac{\pi}{9}$ centered at r_c that contains at least 3 such centers c_1, c_2, c_3. Consider the two senders that are closest to c_r, say c_1 and c_2. Because the cone's angle is $\frac{\pi}{9}$ and $r_i \geq r_c$ for every disk, c_3 must be closer to either c_1 or c_2 than to any point in C. Hence, D_3 violates either the lemma's first or second property. □

The idea behind the proof about the schedule lengths produced by Algorithm 19.1 is to upper-bound the number of links which can prevent a link $f_{ij} \in \mathcal{F}_k$ from being selected for scheduling. Since in every iteration of the while-loop at least one link is scheduled, the number of such preventing links is an upper bound on the time slots used by the **schedule**() procedure before $f_{ij} \in \mathcal{F}_k$ is finally scheduled. We say that a link f_{ab} *blocks* link f_{ij} if the presence of f_{ab} in E_t is the reason why $\mathbf{allowed}(\mathbf{f_{ij}}, \mathbf{E_t})$ evaluates to false. In other words, a link f_{ab} blocks a shorter link f_{ij} if the **schedule**() procedure of Algorithm 19.1 does not schedule f_{ij} simultaneously with f_{ab}. In the sequel, we bound the number of such blocking links for each $f_{ij} \in \mathcal{F}_k$. We begin with a lemma that captures the number of blocking links f_{ab} for which $\tau(f_{ij}) = \tau(f_{ab})$.

Lemma 19.5. *Let B_0 be the set of links $f_{ab} \in \mathcal{F}_k$ that block f_{ij} with $\tau(f_{ab}) = \tau(f_{ij})$. For all f_{ij}, $|B_0| \leq \eta\mu^2$ holds for some constant $\eta < 18$.*

Proof. The proof is based on an area argument. Since all links $f_{ab} \in B_0$ are in the same set of the partition $\mathcal{L}$ as f_{ij}, we know that $\ell(f_{ij}) \leq \ell(f_{ab})$ (every blocking link f_{ab} is considered before f_{ij} in the algorithm) and $\ell(f_{ij}) \geq \frac{1}{2}\ell(f_{ab})$. By the definition of the algorithm, a link f_{ab} is in B_0 if and only if $\mu\ell(f_{ij}) > d(v_i, v_a)$. It follows that the senders of all blocking links for f_{ij} must be located in a disk D_s of radius $\mu\ell(f_{ij})$ around v_i.

Since $\ell(f_{ab}) \geq \ell(f_{ij})$ for all f_{ab} and because the links in $\mathcal{F}_k$ form a nearest neighbor forest, we know that there can be at most one sender of a blocking link in any disk D of diameter $\ell(f_{ij})$. Hence, the number of such disks D required to cover the entire disk D_s constitutes an upper bound on the number of blocking links in B_0. All disks D intersecting D_s are completely inside the disk D'_s, where D'_s has radius $(\mu+\frac{1}{2})\ell(f_{ij})$. Furthermore, the disks D can be tesselated in a grid such that the whole area of D'_s is covered while no point in D'_s is covered by more than two disks D. Hence, defining ρ to be the number of disks D required to cover D'_s, we can write

$$\rho \cdot \left(\frac{1}{2}\ell(f_{ij})\right)^2 \pi \;\leq\; 2 \cdot \left(\left(\mu + \frac{1}{2}\right)\ell(f_{ij})\right)^2 \pi,$$

and by solving for ρ we obtain $\rho \leq 8 \cdot (\mu + 1/2)^2 \leq \eta\mu^2$, which concludes the proof. □

The more intricate part of the analysis is to bound the number of blocking links in other link classes of $\mathcal{L}$. In particular, note that it can be relatively easily shown that there are only a constant number of blocking links for f_{ij} in each link class $L_k \in \mathcal{L}$. However, this bound is not sufficient, as there may be as many as $\Omega(n/\log n)$ different link classes that are considered in the same outer for-loop iteration of the **schedule**() procedure. Hence, we need a much stronger bound in order to guarantee the scheduling complexity as claimed in Theorem 19.11. We start with a helper lemma that characterizes the ratio between the lengths of two links.

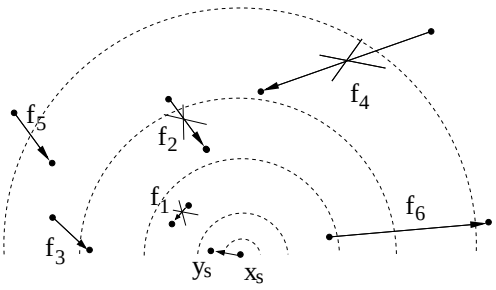

Figure 19.3: Illustration of Algorithm 19.1. In the example, links f_1, f_2, and f_4 are blocking for the link between v_i and v_j. Note that f_6 is not blocking because its receiver is outside the critical ball, even though its sender is close from v_i.

Lemma 19.6. *Let f_{ij} and f_{ab} be two links that are considered in the same subroutine call, and let $\tau(f_{ij}) \geq \tau(f_{ab})$. Then, the length of f_{ab} is at least $\ell(f_{ab}) \geq \frac{1}{2}(3n\beta)^{\delta_{ia}} \cdot \ell(f_{ij})$.*

Proof. By Line 5 of the **schedule**() procedure, only links in length classes $L_j, L_{\log(3\beta n)+j}, L_{2\log(3\beta n)+j}, \ldots$ are considered in the same iteration of this procedure's outer for-loop. The value of δ_{ia} denotes the number of length classes that separate links f_{ij} and f_{ab}, each such separating length class entails at least a doubling of the length. Taking into account that lengths can differ by at most a factor of 2 within a length-class, it follows that $\ell(f_{ab})$ is at least

$$\ell(f_{ab}) \geq \ell(f_{ij}) \cdot 2^{\delta_{ia}\log(3\beta n)-1} = \ell(f_{ij}) \cdot \frac{1}{2}(3n\beta)^{\delta_{ia}}.$$

□

The following key lemma bounds the number of blocking links in other length classes.

Lemma 19.7. *Let B_+ be the set of links $f_{ab} \in \mathcal{F}_k$ that block f_{ij}, with $\tau(f_{ab}) < \tau(f_{ij})$. It holds that for all f_{ij}, $|B_+| \leq (\log_\alpha n + 2)\kappa$, for $\kappa \leq 36$.*

Proof. Recall that the sending node of f_{ij} is v_i and denote all links in B_+ by $f_1, \ldots, f_p$. For all these links f_p, it holds that $\tau(f_{ij}) > \tau(f_p)$, i.e., $\delta_{pi} > 0$. For each such link f_p, s_p and r_p denote its intended sender and receiver, respectively. The links f_p are ordered according to the distance $d(v_i, r_p)$, where f_1 is the link whose r_1 is the closest intended receiver from v_i.

First note that if a link $f_p \in B_+$ blocks f_{ij} and it holds that $d(v_i, r_p) > (3n\beta)^{\frac{\varphi}{\alpha}} \cdot \ell(f_{ij})$, then $\delta_{ip} > \varphi - 1$ must hold and consequently $\delta_{ip} \geq \varphi$. This

is true because if $\delta_{ip} < \varphi$, it holds that $(3n\beta)^{\frac{\delta_{ip}+1}{\alpha}} \cdot \ell(f_{ij}) < d(v_i, r_p)$ and consequently, by the definition of the algorithm, f_p does not block f_{ij}. In combination with Lemma 19.6, this yields the fact that for a dropped link f_p with

$$d(v_i, r_p) \;>\; (3\beta n)^{\frac{\varphi}{\alpha}} \ell(f_{ij}), \tag{19.2}$$

the length $\ell(f_p)$ of the link must be at least

$$\ell(f_p) \;\geq\; \frac{1}{2}(3n\beta)^{\varphi} \cdot \ell(f_{ij}). \tag{19.3}$$

In the following, consider an exponentially growing series of disks C_h, $h = 1, 2, \ldots$ of radius $r_h = (3\beta n)^{\frac{h}{\alpha}} \ell(f_{ij})$ centered at v_i. Furthermore, define a *ring* R_h as the area $C_{h+1} \setminus C_h$, i.e., it holds for every node $v' \in R_h$ that

$$(3\beta n)^{\frac{h}{\alpha}} \ell(f_{ij}) \;<\; d(v_i, v') \;\leq\; (3\beta n)^{\frac{h+1}{\alpha}} \ell(f_{ij}).$$

A key observation for the proof is that there cannot be many links dropped from rings which are close to one another. This intuition is formalized using two helper lemmas. Lemma 19.8 shows that there can only be a constant number of receivers in the first three rings. In Lemma 19.9 we then prove that if for an arbitrary p, the receiver r_p is located in R_h, $h \geq 3$, there cannot be more than κ other intended receivers from dropped links in the subsequent $\alpha(k-1) - 1$ rings.

Lemma 19.8. *It holds that $r_{2\kappa+1}$ is located outside of C_3, i.e., at most 2κ links with receiver in C_3 are blocking for f_{ij}.*

Proof. First, consider all links f_p for which $\delta_{pi} = 1$. Each such link has length at least $\ell(f_p) \geq \frac{1}{2}(3\beta n)\ell(f_{ij})$. Since f_p is blocking, its receiver must be located within distance $(3\beta n)^{2/\alpha}\ell(f_{ij})$ of v_i. For $\alpha > 2$ and large enough n, it holds that $(3\beta n)^{2/\alpha} \leq \frac{1}{2}(3\beta n)$. Assume for contradiction that $\kappa + 1$ or more links f_i with $\delta_{pi} = 1$ exist. Also, draw a disk C of radius $(3\beta n)^{2/\alpha}\ell(f_{ij})$ around v_i, and disks D_p of radius $\ell(f_p)$ around each corresponding sender s_p. Notice that there are at least $\kappa + 1$ disks D_p each of which overlaps with disk C in at least one point (where r_p is located) and no disk D_p contains the center of another disk $D_{p'}$, because the links f_p form a nearest neighbor forest. However, the possibility of packing $\kappa + 1$ disks D_i in such a way contradicts Lemma 19.4 and hence, it follows that there can be at most κ links f_p with $\delta_{pi} = 1$.

Next, we bound the remaining number of blocking links f_p whose receivers r_p are situated in C_3. Each of these remaining links has length at least $\frac{1}{2}(3\beta n)^2\ell(f_{ij})$ because $\delta_{pi} \geq 2$. Moreover, all receivers are located in C_3, that is, $d(v_i, r_p) \leq (3\beta n)^{\frac{3}{\alpha}}\ell(f_{ij}) < \frac{1}{2}(3\beta n)^2\ell(f_{ij})$. Again, it follows by Lemma 19.4 that the number of blocking links with $\delta_{pi} \geq 2$ in C_3 is upper-bounded by κ. ∎

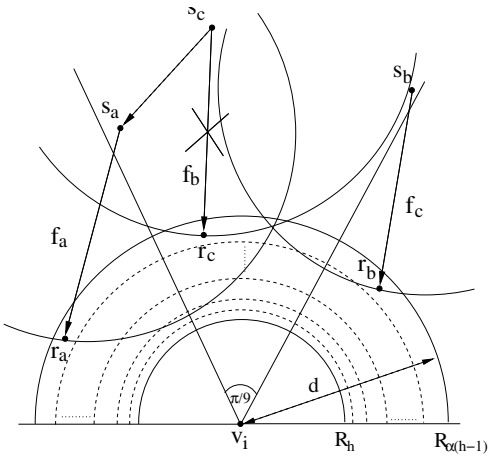

Figure 19.4: Illustration of the proof of Lemma 19.9. Because the length of links f_a, f_b, and f_c is larger than the radius of the disk in which all receivers must be located, at most κ such links can exist. In the example, the closest neighbor of s_c is s_a and not r_c, which yields the contradiction.

Lemma 19.9. *It holds for all p that if $r_p \in R_h$, $h \geq 3$, then $r_{p+\kappa} \in R_{h'}$ for $h' > \alpha(h-1)$. That is, for any $h \geq 3$, there can be at most κ blocking links with receivers in rings $R_h, \ldots, R_{\alpha(h-1)}$.*

Proof. It follows from Equations (19.2), (19.3), and the definition of a ring, that every blocking link with receiver in rings $R_h, \ldots, R_{\alpha(h-1)}$ must be of length at least $\frac{1}{2}(3\beta n)^h \ell(f_{ij})$ (otherwise, it would not be blocking). On the other hand, the distance between a receiver in these rings and v_i is at most

$$\begin{aligned} d(v_i, r_p) &\leq (3\beta n)^{\frac{\alpha(h-1)+1}{\alpha}} \ell(f_{ij}) = (3\beta n)^{h-1+\frac{1}{\alpha}} \ell(f_{ij}) \\ &< \frac{1}{2}(3\beta n)^h \ell(f_{ij}) \leq \ell(f_p), \end{aligned}$$

where the second-to-last inequality holds for $\beta \geq 1$, $\alpha > 2$, and $h \geq 3$. It follows that like in the proof of Lemma 19.8, we can draw a disk D_p with radius $\ell(f_p)$ around each sender s_p in rings $R_h, \ldots, R_{\alpha(h-1)}$. Each of these disks must overlap with the disk centered at v_i of radius $(3\beta n)^{h-1+\frac{1}{\alpha}} \ell(f_{ij})$ and no disk D_p contains the center of another disk. Hence, as illustrated in Figure 19.4, it follows by Lemma 19.4 that there can be at most κ dropped links with receiver in rings $R_h, \ldots, R_{\alpha(h-1)}$. □

Having proven Lemmas 19.8 and 19.9, we can now bound the total number of blocking links in B_+ and thus conclude the proof of Lemma 19.7. By

Lemma 19.8, we know that at most the first 2κ receivers $r_1, \ldots, r_{2\kappa}$ can be located in C_3. All other receivers must be located in a ring R_h for $h \geq 3$. By applying Lemma 19.9, it follows that the receiver $r_{3\kappa+1}$ cannot be closer to v_i than in ring $R_{\alpha(h-1)} = R_{2\alpha}$, receiver $r_{4\kappa+1}$ cannot be closer than in ring $R_{2\alpha^2-\alpha}$, and so forth. By thus recursively applying Lemma 19.9, it follows that receiver $r_{(p+2)\kappa+1}$ cannot be closer than in ring R_{λ_p}, where λ_p is

$$\lambda_p \;=\; 2\alpha^p - \sum_{h=1}^{p-1} \alpha^h \;<\; \alpha^p.$$

Because there are at most n different length classes, the last ring from which a receiver (and its link) can be blocking for f_{ij} is R_n. Consequently, the total number of links that can be blocking for f_{ij} is at most $(p_m + 2)\kappa$, where

$$\alpha^{p_m}(k-2) \;\leq\; n \;\Rightarrow\; p_m \;\leq\; \log_\alpha\left(\frac{n}{k-2}\right) \;\leq\; \log_\alpha n,$$

which implies $|B_+| \leq (\log_\alpha n + 2)\kappa$. □

Lemmas 19.5 and 19.7 provide us a lower bound on the amount of progress achieved by the algorithm when scheduling the links selected in one *phase*. In particular, it allows us to derive a bound on the time required to schedule the nearest neighbor forest in this phase. However, we also need to bound the *number of phases* that the algorithm executes before termination. This is done in the following lemma.

Lemma 19.10. *Let $\mathcal{A}_p$ denote the set of active nodes at the beginning of phase p during the execution of Algorithm 19.1. For each p, it holds that $|\mathcal{A}_{p+1}| \leq |\mathcal{A}_p|/2$.*

Proof. In Line 10 of Algorithm 19.1, all nodes that have an outgoing link (i.e., that need to transmit during this phase) will be removed from $\mathcal{A}$. Consider the connected components of forest $\mathcal{F}_p$. In each such connected component, there is at most 1 node that has no outgoing link, because each connected component forms a directed tree with a unique sink. The claim follows because each connected component consists of at least two nodes. □

Finally, we are ready to prove the main theorem of this section containing the claimed correctness and efficiency results of Algorithm 19.1.

Theorem 19.11. *For every network, Algorithm 19.1 produces a correct schedule $\mathcal{S}$ that induces a strongly-connected subgraph. Furthermore, the length of the schedule is $T(\mathcal{S}) \in O(\log^3 n)$.*

Proof. As for the number of time slots, we start by showing that every call of the **schedule**() subroutine requires at most $O(\log^2 n)$ time slots. By Lemmas 19.5 and 19.7, there are at most

$$B_0 + B_+ \;\leq\; \eta\mu^2 + (\log_\alpha n + 2)\kappa$$

blocking links for each link f_{ij}. Hence, after at most $\eta\mu^2 + (\log_\alpha n + 2)\kappa$ iterations of the while-loop, all links that are considered in the same iteration of the outer for-loop in the **schedule**() subroutine are scheduled for transmission. The number of for-loop iterations being $\lceil \log(3n\beta) \rceil$, it follows that for constant β the number of time slots used by Algorithm 19.1 for scheduling each nearest neighbor forest $\mathcal{F}_p$ is $O(\log^2 n)$.[2]

All that remains to be done to derive the algorithm's scheduling complexity is to bound the number of phases. By Lemma 19.10, the number of active nodes is at least halved in every phase. Therefore, at most $\log n$ phases are required until there remains only a single active node upon which the algorithm terminates. Putting everything together, the algorithm's schedule complexity is

$$T(\mathcal{S}) \in O(\log^3 n).$$

By Lemma 19.3, every transmitted message is successfully received. Furthermore, observe that the union of all scheduled links $\mathcal{F}_p$ forms a directed tree towards a single node (the one node that remains active at the end) in the network. This node can then connect the network with a single transmission in time slot t. Hence, the topology consisting of all scheduled links is connected, i.e., there exists a path between all pairs of nodes. □

Theorem 19.11 implies the following fundamental corollary that bounds the *scheduling complexity of connectivity in wireless networks.*

Corollary 19.12. *The scheduling complexity of a strong connectivity is* $O(\log^3 n)$ *in every wireless network.*

19.2 A Simple Linear-Time Algorithm

The lower bound of Section 18.2 for linear power assignment protocols as well as the structure of Algorithm 19.1 indicate that scheduling is difficult mainly in networks in which some communication links are exponentially longer than others. This raises the question whether the performance of linear power assignment approaches may also deteriorate as badly in case the length of the communication link is less varied, for instance in randomly deployed average-case networks.

In this section, we show that linear power assignment algorithms perform poorly only in scenarios in which there are links belonging to many different orders of magnitude. For simplicity of presentation, we again consider the simple scheduling task Ψ_{min}, i.e., we want that every node can transmit successfully at least once. By applying a technique similar to the one in Section 19.1, the algorithm can be turned into an algorithm for scheduling the *strong connectivity* property at the cost of an additional $O(\log n)$ factor in the scheduling complexity.

[2] Notice that this result implies that our algorithm is capable of scheduling the simple scheduling task Ψ_{min} studied in Chapter 18 in time $O(\log^2 n)$ in every network.

Input: An arbitrarily located set of nodes V
Output: A schedule $\mathcal{S}$ in which every node $v \in V$
can send successfully to its closest neighbor.

For each $v_i \in V$, let f_i be the link to its closest neighbor;
Let $\mathcal{L} = L_0, \ldots, L_{\lfloor \log \mathcal{D} \rfloor}$, where L_k is the set of links with $2^k \leq \ell(f_i) < 2^{k+1}$;
$\mu = 6 \sqrt[\alpha]{\frac{2\beta(\alpha-1)}{\alpha-2}}$; $\rho > 4N$; $t = 1$;
for each $L_k \neq \emptyset$ **do**
  Partition the plane in cells of width $\mu \cdot 2^k$;
  for $j = 1$ **to** 4 **do**
    Select a maximal independent set of cells C_j (cf. Figure 19.5);
    repeat
      For each selected cell C, pick one unscheduled link $f_i \in L_k$ whose intended sender v_i is in C, if there exists such a link;
      $\phi_t(v_i) := \rho \cdot \ell(f_i)^\alpha$;
      Delete f_i from the set of unscheduled links;
      $t = t + 1$;
    until there are no more unscheduled links in active cells;
  end for
end for
$\mathcal{S} := \{\phi_1, \ldots, \phi_{t-1}\}$;

Algorithm 19.2: Linear Power Assignment Algorithm

The *diversity* $g(V)$ of a set of nodes is the number of magnitudes of distances [175]. Formally, $g(V)$ is defined as

$$g(V) := |\{m \mid \exists v_i, v_j \in V : \lfloor \log(d(v_i, v_j)) \rfloor = m\}|.$$

In the example shown in Figure 18.1, for instance, the diversity is $g(V) = \log(2^n) = n$. In our case, $g(V)$ can also be considered as being the number of non-empty length classes of the nearest neighbor forest. In the sequel, we show that Algorithm 19.2 achieves a scheduling complexity of $T(S) \in O(g(V))$ for Ψ_{min}.

The idea of Algorithm 19.2 is simple: simultaneously schedule links of similar length, while guaranteeing a large enough spatial reuse distance between each pair of transmitting nodes. As before, f_i denotes the link from v_i to its closest neighbor, and $\ell(f_i)$ is the length of this link. In each phase of the algorithm, only links belonging to the same *length class* are scheduled. In order to schedule one such phase for links of length $2^k \leq \ell(f_i) < 2^{k+1}$, the algorithm partitions the plane into grid-cells of width $\mu \cdot 2^k$. In each time slot, it chooses a maximal independent set of cells and selects one link in each such cell for scheduling.

The following lemma's proof follows exactly along the lines of the proof of Lemma 19.3 (only using different constants).

Lemma 19.13. *Every node can send successfully in the unique time slot t in which $\phi_t(v_i) > 0$.*

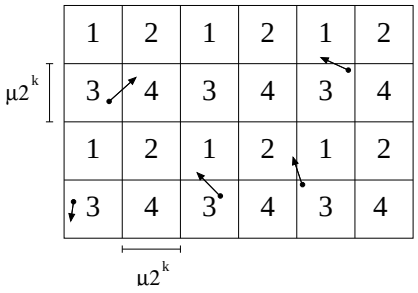

Figure 19.5: In Line 7 of Algorithm 19.2, the algorithm picks all cells numbered by j for some $j = 1, \ldots, 4$. The example shows an inner-loop iteration for length-class L_k and $j = 3$. The algorithm schedules one unscheduled communication link from each selected cell (if there exists one).

Having Lemma 19.13, we obtain the following theorem.

Theorem 19.14. *The schedule $\mathcal{S}$ obtained by Algorithm 19.2 has length at most $T(S) \in O(g(V))$ and fulfills property Ψ_{min}, i.e., each node can send successfully at least once.*

Proof. Correctness follows directly from Lemma 19.13 and from the observation that for every sender v_i, there is a time slot t for which $\phi_t(v_i) > 0$. As for the length of the schedule, observe that there are at most $O(g(V))$ non-empty length classes, i.e., iterations of the outermost loop. Hence, it only remains to prove that a single phase requires only a constant number of time slots.

Consider the phase in which length-class L_k is scheduled. We first show that the number of potential transmitters in a cell can be at most a constant. Because every transmitting node has a link to its *closest* neighbor, the disks D_i of radius $\frac{1}{2}\ell(f_i) \geq 2^{k-1}$ around each transmitter v_i do not overlap. Consider all nodes located in a cell C. The disks D_i belonging to these nodes are completely contained in a square of side-length $(\mu+1) \cdot 2^k$. Hence, it follows from the standard area packing argument that the number of links in C is at most $4(\mu+1)^2 \in O(1)$ in each cell. The proof is now concluded by observing that in a grid, always one fourth of the cells can be scheduled independently as shown in Figure 19.5. □

In combination with the "growing component" technique used in Section 19.1, the following theorem can be derived.

Theorem 19.15. *Algorithm 19.2 can be adapted the strong-connectivity property with scheduling complexity $O(\min(n, g(V) \cdot \log n))$.*

Proof. If $g(V) \cdot \log n < n$, we combine Algorithm 19.2 with the technique of merging clusters iteratively in each phase, as done in Algorithm 19.1.

Using Lemma 19.10, it requires at most $O(\log n)$ phases (each taking time $O(g(V))$) until the scheduled links form a directed tree towards a single node. If $g(V) \cdot \log n \geq n$, the algorithm can simply schedule each node individually. □

Chapter 20

On the Complexity of Arbitrary Topologies

In the previous chapters, we have studied the scheduling complexity of connectivity and other specific request sequences. We have seen that the scheduling complexity of connectivity can be expressed succinctly as a (polylogarithmic) function of the number of nodes n. Is it possible to derive similar results for the scheduling complexity of more general request sequences or topologies? In particular, what can be said about the number of time slots required for successfully scheduling an arbitrary set of communication requests γ_{ij} in a wireless network?

In general, this *scheduling complexity of arbitrary topologies* may not allow for a concise expression as a function of n other than the trivial bound of $O(n)$, which is achieved if nodes are scheduled one after the other. In fact, it is easy to construct examples—as the one depicted in Figure 20.1—with n requests in which the scheduling complexity grows linearly, $\Omega(n)$, even if all sender and receiver pairs are different.

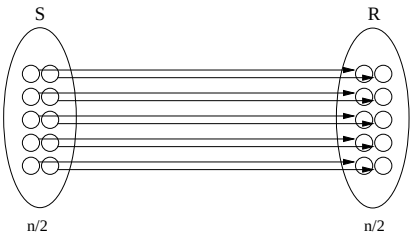

Figure 20.1: The scheduling complexity of the requests in this example is $n/2$. In each time slot, at most one link can be scheduled successfully.

In view of these trivial, existentially tight bounds, it is more interesting to express the scheduling complexity of arbitrary topologies in terms of additional properties besides n. With regard to our discussion of graph-based models versus SINR-based models, it would be particularly intriguing to derive the scheduling complexity of arbitrary requests in dependence of a *graph-theoretic measure* that captures the inherent complexity of scheduling in wireless multi-hop networks. If the scheduling complexity in wireless networks could be expressed by means of a simple, intuitive graph-theoretical measure, this would serve as a legitimation for studying graphs when reasoning about scheduling in wireless networks. In a sense, such a correspondence would thus help bridging the gap between communication and information theoretical SINR models and algorithmic graph models.

One promising graph-theoretical measure for bounding the scheduling complexity of arbitrary request sequences is derived from the field of *topology control*. In a very general sense, topology control in wireless ad hoc and sensor networks can be considered the task of—given a network connectivity graph—computing a subgraph with specific desired properties, such as connectivity, short stretches, sparsity, low interference, or low node degree. Accordingly, there has been considerable research effort towards achieving and combining more and more of these properties [126, 161, 162, 163, 164, 206, 231, 232].

All these approaches have in common, that they model wireless networks as *static graphs*, hence neglecting that, eventually, messages—or, more exactly, radio signals—will have to be sent over these static links in the topologies selected by a topology control algorithm, that is, the static graph of communication links must be *scheduled* on the physical layer. Nonetheless, it turns out that there exists a relationship between a *static interference notion* of topology control [224] and the scheduling complexity. Section 20.1 defines this and related measures.

20.1 Static Interference

The idea of modeling interference in wireless multi-hop networks by means of a static, traffic-independent measure first appeared in [39]. The so-called *in-interference* measure I_{in} of a set of communication requests (that is, a network topology) was later defined in a graph-theoretic context in [224]. This interference measure I_{in} is based on the question of how many other nodes can potentially disturb a given node in the network.[1] With the assumption that nodes use perfectly omnidirectional antennas, the *maximal disk* $\mathcal{D}_i$ of a node v_i represents the transmission range such that all intended receivers of v_i are reached, or, in other words, the disk covering all nodes that are potentially affected by message transmission of v_i to one of its intended receivers.

[1] Similar interference measures have been defined assuming in a sense an antipodal perspective by asking how many other nodes a given node can disturb. It can be shown that the results of the analysis in the remainder of this chapter asymptotically also hold for such I_{out}-interference measures.

Then the interference of a node v_j is defined as the number of other nodes that potentially affect message reception at node v_j:

Definition 20.1. *Given a network topology or a set of communication requests Λ, the in-interference $I_{in}(v_j)$ of a node $v_j \in V$ is defined as*

$$I_{in}(v_j) = |\{v_i | v_i \in V \setminus \{v_j\}, v_j \in \mathcal{D}_i)\}|.$$

In other words, the interference $I_{in}(v_j)$ represents the number of nodes covering v_j with their disks induced by their transmission ranges set to a value large enough to reach all their intended receivers. Note that the in-degree of a node in a given topology Λ does not correspond to its in-interference; the in-degree merely forms a lower bound for its in-interference since it can be "covered" by disks of non-neighboring nodes. The node-level interference defined so far is now extended to an interference measure for a topology or a set of requests Λ:

Definition 20.2. *Given a set of nodes V and a topology or a set of communication requests Λ, the in-interference of Λ is $I_{in}(\Lambda) = \max_{v_i \in V} I_{in}(v_i)$.*

Algorithmically, there are many unresolved questions with regard to static interference minimization problems. The problem is to assign transmission ranges to nodes such that the network becomes connected, but the resulting interference is minimized. In [224], an algorithm for one-dimensional line networks with symmetric edges (i.e., an edge appears in the topology only if *both* nodes have a large enough transmission range) is presented. In particular, the algorithm of [224] achieves an interference of at most $O(\sqrt{n})$. As shown in [224], this is existentially optimal, in the sense that even in the line, there are instances in which every solution generates an interference of $\Omega(\sqrt{n})$. In case of asymmetric, directed edges, the problem becomes somewhat easier. In this case, an algorithm proposed in [96] achieves connectivity with an interference of $O(\log n)$, which is again existentially optimal. As far as minimizing *average interference* with symmetric edges is concerned, an $O(\log n)$-approximation algorithm was proposed in [180], which was shown to be asymptotically optimal in more general metric network instances. Finally, various other notions of interference have been studied in [135], [175], and [177].

It is important to observe, however, that the definition of I_{in} is independent of the SINR model and argues using *circular transmission ranges*, which implicitly assumes a kind of *linear $P \sim d^\alpha$ transmission power policy*, which we have proven to be inefficient for scheduling in Chapter 18. Moreover, it has not been clear—neither from a theoretical nor a practical view—whether the graph-theoretic measures of topology control really bear any significance when it comes to actually scheduling messages in an SINR environment. In this chapter, we demonstrate and prove the existence of fundamental theoretical ties between topology control and the theoretically achievable efficiency of scheduling protocols.

20.2 Algorithm in the Generalized Model

This section proves the following bound on the scheduling complexity of an arbitrary set of requests Λ. In particular, the bound is derived for the generalized physical SINR model as introduced in Section 17.1.

Theorem 20.1. *Given an arbitrary network and a set of communication requests Λ with in-interference I_{in}, all requests $\gamma_{ij} \in \Lambda$ can be successfully scheduled in time $O(I_{in} \cdot \log(n\theta^2) \cdot (\theta^{\frac{4}{\alpha}} + \log n))$. That is, the scheduling complexity $T(\Lambda)$ of a topology or a set of requests Λ with in-interference I_{in} is*

$$T(\Lambda) \in O(I_{in} \cdot \log(n\theta^2) \cdot (\theta^{\frac{4}{\alpha}} + \log n)).$$

We first define some useful notation. If Λ is the set of all communication requests to be scheduled, Λ_i denotes the set of requests for which node v_i is the sender, formally $\Lambda_i = \{\gamma_{ij} \in \Lambda\}$. The set of intended receivers to which a node v_i is supposed to successfully transmit is $R_i = \{v_j \mid \gamma_{ij} \in \Lambda\}$. The Euclidean distance of a node v_i to its most distant intended receiver is called the *radius* r_i of v_i: $r_i = \max_{v_j \in R_i} d(v_i, v_j)$. In case there is no communication request for which node v_i is the sender, that is, if v_i has no intended receiver at all, we define $r_i = 0$.

As it turns out, the scheduling algorithm for connectivity in Chapter 19 already contains most ingredients for the solution for general topologies. In fact, Algorithm 20.1 is a generalization of Algorithm 19.1 in two directions: general topologies and the generalized physical model. In analogy to the connectivity algorithm, Algorithm 20.1 employs a power-assignment policy that favors nodes with small radii over nodes with large radii.

The idea is that we do not schedule individual *links* as in the previous chapter, but we compute the schedule at the level of *nodes* directly. In particular, the algorithm computes a schedule $\mathcal{S}$ in which every node v_i establishes links to *all* its intended receivers R_i in a single time slot by (successfully) *broadcasting* within its radius r_i. In other words, a node is only selected for scheduling by Algorithm 20.1 if all its receivers can successfully decode the message. As different outgoing links of a node v_i may have various lengths, but the required SINR ratio must be fulfilled at each receiver, the *blocking condition* in the $\mathbf{allowed}(\mathbf{v_i}, \mathbf{E_t})$ subroutine must be adequately adapted. Moreover, it no longer makes sense to classify links into length classes as done in Algorithm 19.1. Instead, the algorithm for general topologies partitions the *node-set* V into disjoint sets $\mathcal{S} := (S_0, \ldots, S_{\lfloor \log \Delta \rfloor})$ according to the nodes' radii. The details of these adaptations as well as the necessary alterations for incorporating the generalized physical model are presented in Algorithm 20.1.

Analysis

The analysis proceeds along the lines of the analysis of the connectivity algorithm in Chapter 19. However, the incorporation of the generalized physical model and, particularly, the fact that we deal with general topologies (as

Input: - An arbitrarily located set of nodes V
- A set of communication requests Λ

Output: A schedule $\mathcal{S}$ in which all requests $\gamma_{ij} \in \Lambda$ are scheduled

1: Define two constants ν and μ such that $\nu := 4N$ and $\mu := 1 + 2^{\frac{8}{\alpha}+2} \sqrt[\alpha]{\frac{\beta(\alpha-1)}{\alpha-2}}$; $t := 1$;
2: Partition V into sets $\mathcal{S} = S_0, \ldots, S_{\lfloor \log \mathcal{D} \rfloor}$ such that S_i contains all nodes v_j with $2^i \leq r_j < 2^{i+1}$;
3: Delete all empty sets $S_i \in \mathcal{S}$ and rename $\mathcal{S}$ such that S_i is the i^{th} non-empty set in *decreasing* order of the radii of the contained nodes;
4: **for** $k = 1$ **to** $\lceil \log(3n\beta\theta^2) \rceil$ **do**
5: Let $\mathcal{F}_k$ be the union of all sets $S_{m\lceil \log(3n\beta\theta^2) \rceil + k} \in \mathcal{S}$ for $m \in \mathbb{N}_0$;
6: **for each** $v_i \in \mathcal{F}_k$ **do**
7: $\tau(v_i) := \chi$, where $v_i \in S_\ell$ and S_ℓ is the χ^{th} set in $\mathcal{F}_k$ (in decreasing order of radii);
8: **end for**
9: **while** not all links with intending sender in $\mathcal{F}_k$ have been scheduled **do**
10: $E_t := \emptyset$;
11: Consider all nodes $v_i \in \mathcal{F}_k$ in decreasing order of r_i:
12: **if** *allowed*(v_i, E_t) **then** $E_t := E_t \cup \{v_i\}$;
13: Schedule all $v_i \in E_t$ in time slot t, assigning v_i a transmission power of $P_i = \nu(3n\beta\theta^2)^{\tau(v_i)} \cdot r_i^\alpha$;
14: Remove all scheduled senders ($\mathcal{F}_k := \mathcal{F}_k \setminus E_t$);
15: $t := t + 1$;
16: **end while**
17: **end for**

allowed$(\mathbf{v_i}, \mathbf{E_t})$
1: **for each** $v_j \in E_t$ **do**
2: $\delta_{ij} := \tau(v_i) - \tau(v_j)$;
3: **if** $\tau(v_i) = \tau(v_j)$ and $\mu\theta^{\frac{2}{\alpha}} \cdot r_i > d(v_i, v_j)$ **return false**
4: **else if** $r_i \cdot (3n\beta\theta^2)^{\frac{\delta_{ij}+1}{\alpha}} + r_j > d(v_i, v_j)$ **return false**
5: **end for**
6: **return true**

Algorithm 20.1: Scheduling Algorithm for General Requests

opposed to nearest neighbor forests) raises several subtle details that need to be addressed. In the sequel, those parts of the analysis are highlighted which deviate from the algorithm in Chapter 19.

As in the proof of Algorithm 19.1, the algorithm consists of proving correctness and scheduling complexity. Consider a node v_s that has been selected by Algorithm 20.1 to transmit a message in a given time slot. In analogy to Lemma 19.1, we first bound the interference at an arbitrary receiver $v_r \in R_s$ created by simultaneously transmitting nodes w_i with significantly larger radii. As the set of links to be scheduled is no longer a nearest neighbor

forest as in Lemma 19.1, however, we require a somewhat different proof.

Lemma 20.2. *Consider a time slot t_s in which the algorithm schedules a node v_s for transmission. It holds for all intended receivers $v_r \in R_s$ and for any simultaneously transmitting node $w_i \in V \setminus \{v_s\}$ with $\tau(w_i) < \tau(v_s)$ that*

$$I_r(w_i) \leq \nu\theta(3n\beta\theta^2)^{\tau(v_s)-1}.$$

Proof. Because the radius of w_i must be significantly larger than the radius of v_s, w_i was already in E_t at the time $\mathbf{allowed}(\mathbf{v_s}, \mathbf{E_t})$ evaluated to true and the algorithm selected node v_s for scheduling. Consequently, the distance $d(v_s, w_i)$ must have been at least

$$d(v_s, w_i) \geq r_s \cdot (3n\beta\theta^2)^{\frac{\delta_{si}+1}{\alpha}} + r_i > r_s + r_i,$$

where r_i is w_i's radius, and therefore $d(v_r, w_i) > r_i$. The interference caused by w_i at v_r is consequently at most

$$\begin{aligned} I_r(w_i) &\leq \theta \cdot \frac{P_i}{d(v_r, w_i)^\alpha} \leq \frac{\theta\nu(3n\beta\theta^2)^{\tau(w_i)} r_i^\alpha}{r_i^\alpha} \\ &= \nu\theta(3n\beta\theta^2)^{\tau(w_i)} \leq \nu\theta(3n\beta\theta^2)^{\tau(v_s)-1}, \end{aligned}$$

which concludes the proof. □

Next, we bound the interference created by transmitting nodes with smaller radii. Recall that these nodes transmit at a (relatively speaking) increased power level and can therefore disturb potential receivers that are distant, relative to their own radius.

Lemma 20.3. *Consider a time slot t_s in which the algorithm schedules a node v_s for transmission. It holds for all intended receivers $v_r \in R_s$ and for any simultaneously transmitting node $w_i \in V \setminus \{v_s\}$ with $\tau(w_i) > \tau(v_s)$ that*

$$I_r(w_i) \leq \nu\theta(3n\beta\theta^2)^{\tau(v_s)-1}.$$

Proof-Sketch. The proof is analogous to the one given in Lemma 19.2 (albeit for the adjusted version of the $\mathbf{allowed}(\mathbf{v_i}, \mathbf{E_t})$ subroutine) and we therefore limit ourselves to a sketch. The interference $I_r(w_i)$ incurred by a node w_i at v_r is at most

$$I_r(w_i) \leq \theta \cdot \frac{\nu(3n\beta\theta^2)^{\tau(w_i)} \cdot r_i^\alpha}{d(w_i, v_r)^\alpha}.$$

Assuming for contradiction that there exists a node w_i with $\tau(w_i) > \tau(v_s)$ and $I_r(w_i) > \nu\theta(3n\beta\theta^2)^{\tau(v_s)-1}$, it can be shown that the distance between w_i and v_r is at most $r_i \cdot \sqrt[\alpha]{(3n\beta\theta^2)^{\tau(w_i)-\tau(v_s)+1}}$ and therefore

$$\begin{aligned} d(w_i, v_s) &< \sqrt[\alpha]{(3n\beta\theta^2)^{\tau(w_i)-\tau(v_s)+1}} \cdot r_i + r_s \\ &= r_i \cdot (3n\beta\theta^2)^{\frac{\delta_{is}+1}{\alpha}} + r_s \end{aligned}$$

must hold. This contradicts the fact that w_i and v_s are selected for scheduling in the same time slot. □

Finally, it remains to prove a bound on the interference created by simultaneously sending nodes in the same set of the partition.

Lemma 20.4. *Consider a time slot t_s in which the algorithm schedules a node v_s for transmission. It holds for all intended receivers $v_r \in R_s$ and for any simultaneously transmitting node $w_i \in V \setminus \{v_s\}$ with $\tau(w_i) = \tau(v_s)$ that*

$$I_r(w_i) \leq \nu\theta(3n\beta\theta^2)^{\tau(v_s)-1}.$$

Proof. The proof is analogous to the one given in Lemma 19.3. Let $\mathcal{T}$ denote the set of simultaneously transmitting nodes w_i with $\tau(w_i) = \tau(v_s)$. By the definition of $\mathbf{allowed}(\mathbf{w_j}, \mathbf{E_t})$ a node $w_i \in \mathcal{T}$ prevents all nodes $w_j \in \mathcal{T}$ for which $\mu\theta^{\frac{2}{\alpha}} \cdot r_j > d(w_i, w_j)$ from being added to $\mathcal{T}$.

This allows us to set up the standard area argument. In particular, disks D_i of radius $\frac{1}{4}\mu\theta^{\frac{2}{\alpha}} r_s$ centered at every node $w_i \in \mathcal{T}$ do not overlap. Therefore, we can place rings R_λ of width $\mu\theta^{\frac{2}{\alpha}} r_s$ around v_s and bound the cumulated interference created by senders in given ring, individually. Formally, let R_λ contain all nodes $w_i \in \mathcal{T}$ for which $(\lambda - \frac{1}{2})\mu\theta^{\frac{2}{\alpha}} r_s < d(v_s, w_i) \leq (\lambda + \frac{1}{2})\mu\theta^{\frac{2}{\alpha}} r_s$. All disks D_i of transmitters in some ring R_λ are located entirely in an extended ring of area

$$A(R_\lambda^+) = \left[\left(\left(\lambda + \frac{3}{4}\right)\mu\theta^{\frac{2}{\alpha}} r_s\right)^2 - \left(\left(\lambda - \frac{3}{4}\right)\mu\theta^{\frac{2}{\alpha}} r_s\right)^2\right]\pi = 3\lambda\mu^2\theta^{\frac{4}{\alpha}} r_s^2\pi.$$

Each transmitter in R_λ to v_r has a distance of at least $((\lambda - \frac{1}{2})\mu\theta^{\frac{2}{\alpha}} - 1)r_s$ from v_r. Moreover, the transmission power of each such node is no more than $\nu(3n\beta\theta^2)^{\tau(v_s)} \cdot (2r_s)^\alpha$. The total interference I_λ generated by nodes in ring R_λ is therefore at most

$$I_\lambda \leq \frac{A(R_\lambda^+)}{A(D_i)} \cdot \frac{\theta\nu(3n\beta\theta^2)^{\tau(v_s)} \cdot (2r_s)^\alpha}{(((\lambda - \frac{1}{2})\mu\theta^{\frac{2}{\alpha}} - 1)r_s)^\alpha} \leq \frac{48\nu(3n\beta\theta^2)^{\tau(v_s)} 2^{2\alpha}}{\theta\lambda^{\alpha-1}(\mu-1)^\alpha}.$$

Summing up the interference over all rings R_λ, we obtain

$$\begin{aligned}
\sum_{\lambda=1}^{\infty} I_\lambda &\leq \frac{48\nu(3n\beta\theta^2)^{\tau(v_s)} 2^{2\alpha}}{\theta(\mu-1)^\alpha} \sum_{\lambda=1}^{\infty} \frac{1}{\lambda^{\alpha-1}} \\
&< \frac{48\nu(3n\beta\theta^2)^{\tau(v_s)} 2^{2\alpha}}{\theta(\mu-1)^\alpha} \cdot \frac{\alpha-1}{\alpha-2} \\
&< \frac{\nu}{\theta} \cdot (3\beta)^{\tau(v_s)-1} \cdot (n\theta^2)^{\tau(v_s)}.
\end{aligned}$$

□

Based on Lemmas 20.2, 20.3, and 20.4, it is now easy to derive the correctness result.

Theorem 20.5. *For every request $\gamma_{sr} \in \Lambda$, there exists a unique time slot t_s in which v_r successfully receives a message from v_s.*

Proof. Consider a node v_s that is scheduled for transmission in a give time slot t_s. When summing up the total interference created by simultaneously transmitting nodes bounded in Lemmas 20.2, 20.3, and 20.4, we obtain

$$\begin{aligned} I_r &\leq \frac{\nu}{\theta}(3\beta)^{\tau(v_s)-1}(n\theta^2)^{\tau(v_s)} + \nu\theta(3\beta\theta^2)^{\tau(v_s)-1}n^{\tau(v_s)} \\ &= \frac{2\nu\theta}{3} \cdot (\beta\theta^2)^{\tau(v_s)-1} \cdot (3n)^{\tau(v_s)}. \end{aligned}$$

The SINR at *any* intended receiver $v_r \in R_s$ is therefore at least

$$SINR_r \geq \frac{\frac{\nu}{\theta}(3n\beta\theta^2)^{\tau(v_s)}}{N + \frac{2}{3}\nu\theta(\beta\theta^2)^{\tau(v_s)-1} \cdot (3n)^{\tau(v_s)}},$$

which can be shown to be larger than β by plugging in the definitions of ν and $n, \beta \geq 1$. From this, it follows that if v_s is scheduled in Line 13 of Algorithm 20.1, all its receivers $v_r \in R_s$ receive the message from v_s, i.e., v_s can correctly broadcast to all nodes within radius r_s.

As in the proof of Lemma 19.3, the proof is concluded by observing that there is a unique time slot for every node in which it can transmit. □

When it comes to the actual time complexity of Algorithm 20.1, the proof deviates from the corresponding part of Chapter 19, because we can no longer argue solely about nearest neighbor forests. Instead, the following simple lemma provides a relationship between the network's geometric formation and its in-interference.

Lemma 20.6. *In any disk D of diameter d, there can be at most $I_{in} + 1$ nodes v_i with $r_i \geq d$.*

Proof. Assume for contradiction that there are $Q > I_{in} + 1$ such nodes in the disk. Since $r_i \geq d$ for all v_i, the disk of radius r_i around each node covers the entire disk D. Hence, the interference experienced by each node in D is at least $Q - 1$, which contradicts the definition of I_{in} if $Q > I_{in} + 1$. □

In the sequel, we bound the number of *blocking nodes* (analogous to the notion of blocking links in Chapter 19) of a node v_s.

Lemma 20.7. *Let B_0 be the set of nodes $w_i \in V$ that block v_s with $\tau(w_i) = \tau(v_s)$. For all v_s, $|B_0| \leq \eta\mu^2\theta^{\frac{4}{\alpha}}(I_{in} + 1)$ holds for some constant $\eta < 18$.*

Proof-Sketch. The proof is again based on an area argument. Since all nodes $w_i \in B_0$ are in the same set of the partition $\mathcal{S}$ as v_s, it holds that $\frac{1}{2}r_i \leq r_s \leq r_i$. Moreover, all blocking nodes for v_s must be located in a disk D_s of radius $\mu\theta^{\frac{2}{\alpha}}r_s$ around r_s. By Lemma 20.6, we know that there can be at most $I_{in} + 1$ blocking nodes in any disk D of diameter r_s. Hence, the number of

such disks D required to cover the entire disk D_s times $(I_{in}+1)$ constitutes an upper bound on the number of blocking nodes in B_0. The corresponding calculation is equivalent to the one in the proof of Lemma 19.5. □

For the purpose of bounding the number of blocking nodes with larger radii, we have to establish a relationship between the radii of different nodes. The next lemma therefore corresponds directly to Lemma 19.6 and the proof is analogous.

Lemma 20.8. *Let v_i and v_j be two nodes that are considered in the same iteration of the for-loop, and let $\tau(v_i) \leq \tau(v_j)$. Then, for $\delta_{ij} = \tau(v_i) - \tau(v_j)$, it holds that $r_i \geq \frac{1}{2}(3n\beta\theta^2)^{\delta_{ij}} \cdot r_j$.*

For the next couple of lemmas, we need to introduce some additional notation. Specifically, we define the *reduced distance* ζ_s^i of w_i from v_s to be $\zeta_s^i = d(v_s, w_i) - r_i$. In words, the reduced distance is a lower bound on the minimum possible distance between v_s and an intended receiver of w_i. Note that in procedure **allowed**$(\mathbf{v_i}, \mathbf{E_t})$ node v_s is blocked by a node $w_i \in E_t$, $\tau(v_s) > \tau(w_i)$, if and only if $r_s(3n\beta\theta^2)^{\frac{\delta_{si}+1}{\alpha}} > \zeta_s^i$.

Lemma 20.9. *For any $\varphi \geq 2$ and $\sigma = 72$, there can be at most $\sigma(I_{in}+1)$ blocking nodes w_i for a node v_s with reduced distance*

$$(3n\beta\theta^2)^{\frac{\varphi}{\alpha}} \cdot r_s \;<\; \zeta_s^i \;\leq\; (3n\beta\theta^2)^{\varphi} \cdot r_s.$$

Proof. Assume for contradiction that there exists a set of $\sigma(I_{in}+1)+1$ or more nodes w_i that are blocking v_s with each ζ_s^i in the range specified in the lemma. Denote this set of nodes by $\mathcal{B} \subseteq V$. First note that if a node $w_i \in \mathcal{B}$ blocks v_s and the reduced distance is $\zeta_s^i > (3n\beta\theta^2)^{\frac{\varphi}{\alpha}} \cdot r_s$, then $\delta_{is} > \varphi - 1$ must hold and consequently $\delta_{is} \geq \varphi$. This is true because if $\delta_{is} < \varphi$, it holds that $(3n\beta\theta^2)^{\frac{\delta_{si}+1}{\alpha}} \cdot r_s < \zeta_s^i$ and consequently, by the definition of the algorithm, w_i does not block v_s. Hence, in combination with Lemma 20.8, we know that all blocking nodes $w_i \in \mathcal{B}$ have a radius of at least

$$r_i \;\geq\; \frac{1}{2}(3n\beta\theta^2)^{\varphi} \cdot r_s. \tag{20.1}$$

We now show that if $|\mathcal{B}| \geq \sigma(I_{in}+1)+1$, then there must be a node that has in-interference at least $I_{in}+1$, which leads to a contradiction. For this purpose we consider a transformation $\mathcal{B}'$ of the node set $\mathcal{B}$ which does not increase I_{in}. We then show that in this transformed instance $\mathcal{B}'$, in-interference is too high.

Consider the following transformation of the node set $\mathcal{B}$ into a node set $\mathcal{B}'$: We replace each node $w_i \in \mathcal{B}$ with radius r_i by a node $w_i' \in \mathcal{B}'$ with radius $r_i' = \frac{1}{2}(3n\beta\theta^2)^{\varphi} \cdot r_s$. Specifically, node w_i' is located on the straight line connecting v_s and w_i at distance $\zeta_s^i + \frac{1}{2}(3n\beta\theta^2)^{\varphi} \cdot r_s$ from v_s, as shown in Figure 20.2. Note that the disk with radius r_i' centered at w_i' is entirely contained in the disk with radius r_i around w_i (cf. Inequality (20.1)); this

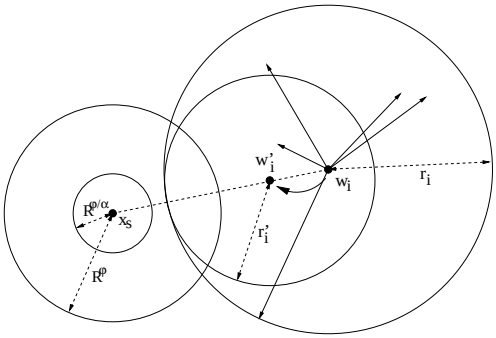

Figure 20.2: Example for the transformation used in the proof of Lemma 20.9. with $R^{\varphi/\alpha} = (3n\beta\theta^2)^{\frac{\varphi}{\alpha}} \cdot r_s$ and $R^{\varphi} = (3n\beta\theta^2)^{\varphi} \cdot r_s$, respectively. The disk with center w_i and radius r_i is replaced by the smaller disk with center w_i' and radius r_i', such that $\zeta_s^i = \zeta_s^{i'}$. The transformation does not increase I_{in}.

transformation cannot increase the in-interference of any node in the network. Moreover, because $d(v_s, w_i)$ and r_i are reduced by the same amount, this transformation does not change the value ζ_s^i for any transformed node, that is, $\zeta_s^i = \zeta_s^{i'}$ for any $w_i \in \mathcal{B}$ and its transformation $w_i' \in \mathcal{B}'$. Finally, note that transforming $\mathcal{B}$ to $\mathcal{B}'$ is always possible.

We now look at the in-interference of node set $\mathcal{B}'$. According to the second inequality of the Lemma to be proven, the reduced radius ζ_s^i of each node $w_i' \in \mathcal{B}'$ is at most $(3n\beta\theta^2)^{\varphi} \cdot r_s$. All radii being $r_i' = \frac{1}{2}(3n\beta\theta^2)^{\varphi} \cdot r_s$, it follows that all nodes $w_i' \in \mathcal{B}'$ are located in a disk D_s^{φ} of radius $\frac{3}{2}(3n\beta\theta^2)^{\varphi} \cdot r_s$ centered at v_s.

Consider disks D of radius $\frac{1}{4}(3n\beta\theta^2)^{\varphi} \cdot r_s$. By the standard area argument also applied in the proof of Lemma 19.5, the number of disks ρ required to cover the entire disk D_s^{φ} is at most

$$\rho \;\leq\; 2 \cdot \frac{(\frac{3}{2}(3n\beta\theta^2)^{\varphi} r_s)^2 \pi}{(\frac{1}{4}(3n\beta\theta^2)^{\varphi} r_s)^2 \pi} \;=\; \sigma.$$

Since by assumption there are at least $\sigma(I_{in}+1)+1$ nodes in $\mathcal{B}'$, there must exist a disk D that contains at least $I_{in}+2$ of these nodes. This, however, establishes a contradiction because by Lemma 20.6 there can be at most $I_{in}+1$ nodes $w_i' \in \mathcal{B}'$ in any disk D of diameter $d = \frac{1}{2}(3n\beta\theta^2)^{\varphi} r_s$. □

Lemmas 20.8 and 20.9 yield the following key lemma.

Lemma 20.10. *Let B_+ be the set of nodes $w_i \in V$ that block v_s, with $\tau(w_i) < \tau(v_s)$. It holds that for all v_s that $|B_+| \leq (\log_{\alpha} \frac{n}{\alpha} + 1)\sigma(I_{in}+1)$, for σ as defined in Lemma 20.9.*

Proof. By Lemma 20.9, we know that there can be at most $\sigma(I_{in}+1)$ blocking nodes w_i for v_s with reduced distance

$$(3n\beta\theta^2)^{\frac{\varphi}{\alpha}} \cdot r_s \;<\; \zeta_s^i \leq (3n\beta\theta^2)^{\varphi} \cdot r_s.$$

In particular, this means that there are at most $\sigma(I_{in}+1)$ blocking nodes with reduced distance $(3n\beta\theta^2)^{\frac{1}{\alpha}} \cdot r_s \;<\; \zeta_s^i \leq (3n\beta\theta^2) \cdot r_s$, at most $2\sigma(I_{in}+1)$ such nodes with $(3n\beta\theta^2)^{\frac{1}{\alpha}} \cdot r_s \;<\; \zeta_s^i \;\leq\; (3n\beta\theta^2)^{\alpha} \cdot r_s$, and so forth. More generally, there are at most $\kappa\sigma(I_{in}+1)$ blocking nodes with

$$(3n\beta\theta^2)^{\frac{1}{\alpha}} \cdot r_s \;<\; \zeta_s^i \;\leq\; (3n\beta\theta^2)^{\alpha^{\kappa-1}} \cdot r_s.$$

Because the partition $\mathcal{S}$ consists of at most n non-empty sets S_i, it holds that $\tau(v_s) - \tau(w_i) \leq n$ for all w_i. Therefore, the reduced distance of any blocking node cannot exceed $r_s(3n\beta\theta^2)^{\frac{n}{\alpha}}$. For $\kappa = \log_\alpha \frac{n}{\alpha} + 1$, on the other hand, the reduced distance is at most $r_s(3n\beta\theta^2)^{\alpha^{\log_\alpha \frac{n}{\alpha}}} = r_s(3n\beta\theta^2)^{\frac{n}{\alpha}}$. Hence, there can be at most $(\log_\alpha \frac{n}{\alpha} + 1)\sigma(I_{in}+1)$ blocking nodes for v_s, from which the lemma follows. □

Finally, we can put everything together in the following theorem, which implies Theorem 20.1.

Theorem 20.11. *The number of time slots required by Algorithm 20.1 to successfully schedule all links $\gamma_{ij} \in \Lambda$ is at most $O\left(I_{in} \cdot \log(n\theta^2)(\theta^{\frac{4}{\alpha}} + \log n)\right)$.*

Proof. By Lemmas 20.7 and 20.10, there are at most

$$B_0 + B_+ \;\leq\; \eta\mu^2\theta^{\frac{4}{\alpha}}(I_{in}+1) + (\log_\alpha \frac{n}{\alpha} + 1)\sigma(I_{in}+1)$$

blocking nodes for each node v_s. Hence, after at most $\eta\mu^2\theta^{\frac{4}{\alpha}}(I_{in}+1) + (\log_\alpha \frac{n}{\alpha} + 1)\sigma(I_{in}+1) + 1$ iterations of the while-loop, all nodes that are considered in the same iteration of the outer for-loop are scheduled for transmission. The theorem follows because the number of for-loop iterations is $\lceil \log(3n\beta\theta^2) \rceil$ and β is a constant. □

In the standard physical model ($\theta = 1$) the scheduling complexity of Algorithm 20.1 reduces to $O(I_{in} \cdot \log^2 n)$.

Discussion

Having displayed the significance of static in-interference in the previous section, it is interesting to more closely specify the value I_{in} for the most basic network property, that is connectivity. In particular, we distinguish between *strong connectivity with directed links* and *connectivity with undirected links*, as it turns out that requesting links to be symmetric significantly complicates the task of quickly scheduling communication requests.

In the following, we first take a look at connected topologies with *asymmetric* (also called directed or unidirectional) links. In particular, we consider strongly connected topologies, meaning that there exists from every node v_i in the network to every other node v_j a path containing only links oriented from v_i to v_j. In [96], the following theorem has been proven.

Theorem 20.12 ([96]). *Given an arbitrary set of nodes V, there exists a strongly connected topology with asymmetric links having in-interference $I_{in} \in O(\log n)$. On the other hand, there exist networks for which every strongly connected topology with asymmetric links has interference $I_{in} \in \Omega(\log n)$.*

It is often argued that communication over asymmetric wireless links is costly or in general unacceptably cumbersome; not even simple acknowledgement of a transmitted packet is easily possible over an asymmetric link. From a scheduling perspective, however, demanding communication links to be symmetric does not come for free. Specifically, it has been shown in [224] that the in-interference experienced at a node if links are required to be symmetric can be as high as $\Omega(\sqrt{n})$. Notice that this is significantly higher than the $O(\log n)$ interference bound that holds in the case where links can be asymmetric.

Theorem 20.13 ([224]). *Given an arbitrary set of nodes V, there exist networks for which every connected topology with symmetric links exhibits an in-interference of at least $I_{in} \in \Omega(\sqrt{n})$.*

On the other hand, we present a simple randomized algorithm that achieves an in-interference of $O(\sqrt{n \log n})$ in every network.

Theorem 20.14. *Given an arbitrary set of nodes V, Algorithm 20.2 computes a connected topology with symmetric links with in-interference at most $I_{in} \in \Omega(\sqrt{n \log n})$.*

Algorithm executed at every node $v \in V$:
1: Choose to become a *hub* randomly and independently with probability $p_h = \sqrt{\log n / n}$;
2: **if** v has chosen itself to become a *hub* **then**
3: Set transmission range such that all nodes in V can be reached;
4: **else**
5: Set transmission range to the nearest *hub*;
6: **end if**

Algorithm 20.2: Low In-Interference with Symmetric Links

Proof. Since Algorithm 20.2 assigns transmission ranges instead of selecting single links, we define the computed topology to contain all possible symmetric links that are realizable—in accordance with the definition of I_{in}—with the assigned transmission ranges. In other words, every symmetric link

(v_i, v_j) belongs to the resulting topology if the transmission radii of both v_i and v_j are at least $d(v_i, v_j)$. Consequently—since every hub covers all other nodes—the hubs form a clique. And as every non-hub is connected with its nearest hub (again via a symmetric link), the whole resulting topology is connected.

As for the upper bound, consider an arbitrary node $v \in V$. Node v may experience interference from all hub nodes. Using standard Chernoff bounds, it can be shown that the total number of hubs elected in the network is at most $2\sqrt{n \log n}$ with probability greater than $1 - \frac{1}{n^2}$. It remains to be shown that the total interference induced by non-hubs is also at most $O(\sqrt{n \log n})$ with high probability. Assume for contradiction that node v is covered by more than $12\sqrt{n \log n}$ non-hub nodes. Partition the area around v into 6 cones of angle $\pi/3$ each (such that the border line between two cones is defined to be part of its neighboring cone in positive rotational direction). Since there are more than $12\sqrt{n \log n}$ non-hubs interfering with v, there must exist a cone C in which there are at least $2\sqrt{n \log n} + 1$ of these nodes. Let V_C be the set of non-hubs interfering with v in cone C and let v_{max} be a node in V_C with maximal distance from v. By the definition Algorithm 20.2, the transmission range of v_{max} reaches only to its *nearest* hub. Because the cone angle is $\pi/3$, it holds that $d(v_{max}, v) > d(v_{max}, w)$ for all $w \in V_C \setminus \{v_{max}\}$. Differently put, there are at least $2\sqrt{n \log n}$ non-hubs closer to v_{max} than v. If one of these non-hubs had become a hub, the transmission range of v_{max} would not cover v. The probability P_{none} that none of these $2\sqrt{n \log n}$ nodes becomes a hub is

$$P_{none} \;=\; \left(1 - \sqrt{\frac{\log n}{n}}\right)^{2\sqrt{n \log n}} \;\leq\; e^{-2 \log n} \;=\; \frac{1}{n^2}.$$

The probability that any of the n nodes is covered by more than $12\sqrt{n \log n}$ non-hubs is therefore at most $n \cdot 1/n^2$, and the probability that this holds for no node at all is consequently at least $1 - 1/n$. Combining this with the upper bound on the number of hubs, the theorem follows. □

A comparative interpretation of Theorems 20.12 and 20.13 therefore leads to the conclusion that the scheduling of a connected topology with exclusively *symmetric links* is by its nature significantly more costly than the scheduling of a connected topology using *asymmetric links*. In a sense, this forms an antithesis to the often made assumption that the use of symmetric links is mandatory for practical reasons. At least, this observation provides a new argument to the discussion whether asymmetric edges are valid to be considered for network forming or if they ought to be disregarded altogether.

Chapter 21

Conclusions and Outlook

This part of the thesis was aimed at studying communication models that more closely capture the nature of signal propagation in wireless networks. When studying *algorithmic aspects* of scheduling in a SINR-based physical model, it becomes obvious that even simple physical phenomena are not adequately modeled by standard graph models. In fact, there are problem settings in which graph models inevitably deviate from the physical reality to such a large degree that theoretical bounds and results derived in graph models become almost irrelevant for practical purposes.

It is therefore interesting to gain an understanding of the fundamental possibilities and limitations of low-level network tasks such as scheduling in more realistic wireless communication models. As it has turned out, studying algorithmic aspects of scheduling in wireless networks on the level of physical SINR models reveals previously unknown and practically relevant aspects of wireless communication. Moreover, studying lower abstraction layers also yields challenging algorithmic problems that demand for novel techniques and methods. Because the constraints implied by the SINR model are inherently *non-linear*, for instance, standard optimization techniques based on linear programs—which typically work in graph-based models, e.g. [154]—fail when studying scheduling in physical models. Therefore, although looking somewhat arduous as an algorithmic model, focusing on low-level aspects of wireless networks turns out to be interesting and insightful from this point of view.

In Chapter 18, we have shown that standard and frequently assumed power assignment schemes cannot be competitive even when it comes to simple communication requests. This sub-optimality of uniform and linear $P \sim d^\alpha$ power assignment schemes may have an impact on the way certain problems in wireless networks are modeled and analyzed. The inefficiency of any scheduling protocol based on linear $P \sim d^\alpha$ power assignment, for instance, affects a large body of theoretical work on energy-efficient wireless network design. In particular, concepts such as *minimum energy broadcast*, *minimum energy paths*, or in general *energy-cost metrics* may have to be

reconsidered, because if a protocol actually assigns power levels according to such energy-costs, the resulting schedules would be inherently slow compared to the optimum.

It is interesting to discuss the connections between our results on the scheduling complexity and known bounds on the *capacity* of wireless networks [117]. The results on the capacity of wireless networks essentially give a negative answer to the possibilities of wireless networks by limiting the maximal *throughput* that can be achieved per node as the number of nodes in the network grows. Specifically, the throughput per node decreases by a factor of roughly $1/\sqrt{n}$ as n increases. In contrast, our result is of a more positive nature. Specifically, the algorithm of Chapter 19 demonstrates that by using a proper power assignment scheme, complex communication requests can theoretically be scheduled efficiently even in very large networks. This implies that when it comes to actually *scheduling* transmissions in a wireless network, there exists no fundamental scaling problem as exhibited in the study of capacity.

There remains a wide range of directions for future research. Most obviously, one could investigate the scheduling complexity of other specific network topologies. Possibly a more fundamental question, however, is related to the question of *lower bounds*. Currently, no non-trivial lower bound on the scheduling complexity in wireless networks is known. It would be interesting if we could prove a lower bound of $\Omega(I_{in})$ on the scheduling complexity of arbitrary topologies, because this would imply that our algorithm in Chapter 20 is competitive relative to an optimal scheduling algorithm. Coming up with a non-constant general lower bound on the scheduling complexity in the SINR model is challenging, because any algorithm in the SINR model inherently has "two degrees of freedom": scheduling and power assignment. In combination with the fact that SINR constraints are inherently non-linear, this renders lower bound proofs difficult. One possible step towards general lower bounds could be to generalize the lower bounds for uniform and linear power assignment algorithms to a wider, yet still restricted class of algorithms.

Another potential direction for future research is to study combined *routing and scheduling problems* in SINR models from a *worst-case algorithmic point of view*. In this problem, we are given either a number of requests of the form (s_i, d_i) consisting of source and destination pairs, or alternatively, a number of paths p_i, that have to be scheduled as quickly as possible. Ideally, one could devise routing algorithms with a performance guarantee relative to an optimal routing algorithm, or relative to (adequately adapted) parameters dilation D and congestion C, as done in traditional work on routing [160]. In wireless networks, problems of this kind have been studied only in (restricted) graph-based models. Considering the drastic inconsistencies between graph-based and SINR-based models when it comes to scheduling, however, these graph-based results may divert largely from the physically achievable bounds. Again, the main algorithmic problem that must be overcome is to bound the optimum (or, equivalently, the congestion C), that is, to derive a lower bound on the scheduling complexity.

From a more practical point of view, it will be interesting to turn the

theoretical findings of the previous chapters into practical MAC layer and scheduling protocols. Ultimately, the assignment of non-linear power levels to nodes could help in developing more efficient network protocols. For a simple example, consider a *data gathering* application with high throughput requirements in *heterogeneous wireless multi-hop networks*. In such networks, there are energy-restricted wireless nodes that gather data and locally distribute or forward this data for aggregation, as well as a few designated, more powerful nodes. Eventually, the data has to be sent to a base station, a task which is preferably done by the long-range nodes, instead of regular sensor nodes. In any graph-based model, local communication among regular nodes and long-range communication among designated nodes must be coordinated (either in the time or frequency space, or by using spatial multiplexing). As in the four-node example of Figure 17.2, however, long-range and short-range communication can *coexist*, that is, regular nodes can communicate with each other *while* long-range nodes send data to the base station. In other words, we could devise efficient data gathering applications based on the fact that simultaneous short-range and long-range transmission does not lead to collisions. This could result not only in higher throughput, but also in a significantly smaller coordination overhead between different regions of the networks.

Finally, the *physical model* and its generalized companion as introduced in Chapter 17 are in many ways still highly idealized. On the one hand, these models do not adequately account for obstacles, and on the other hand, the physical model by itself represents a rather simple—some would say "naive"—channel fading model. In the networking literature, much more realistic channel models have been proposed and it would be interesting to plunge deeper into the *algorithmic nature* of communication in these even more realistic and physical models.

Part IV

Selfishness in Networks

Chapter 22

Selfishness in Networks

In Part I of this thesis, we have argued that the absence of global knowledge and the resulting need for local computation are key challenges when dealing with modern large-scale and highly decentralized distributed systems. As already pointed out in the conclusion of Part I, however, there is another fundamental characteristic that distinguishes networks such as the Internet, peer-to-peer networks or mobile ad hoc networks from traditional distributed systems. The nodes of these computer systems are typically governed by socio-economic agents whose main interest is not the optimization of the network's entirety, but rather the maximization of their own benefit. Moreover, there may not be a central authority that designs the system and controls that every participating agent behaves in a benevolent, globally coordinated manner. In other words, many of today's most interesting and algorithmically most challenging large-scale networks are characterized by the fact that each node or agent will—like companies in free economy—strive for optimizing its *own* benefit or reduce its own cost, regardless of its actions' impact on the global social welfare.

In his influential survey [190], Papadimitriou has argued that the Internet has surpassed the von Neumann computer as the most complex computational artifact of our time. In particular, he pointed out that the Internet has a *socio-economic* complexity whose understanding crucially requires techniques from mathematical economics and game theory. And indeed, by modeling the players as utility-maximizing agents, the study of network problems using game theoretic techniques has provided great insights into selfish behavior on all layers of distributed systems in recent years.

One of the most fundamental and frequently studied notions of rationality is the so-called *Nash equilibrium*. Intuitively, a Nash equilibrium constitutes a situation in a game in which no player has an incentive to change its current strategy given that all other players remain with their strategy. In this sense, a Nash equilibrium defines a point of stability in the system, because no player can unilaterally improve its situation by changing its strategy. While the *social optimum* (so far simply denoted by "the optimum" in this thesis)

is the outcome of a global optimization process, a Nash equilibrium may be the outcome of a process in which selfish, rational agents try to optimize their own individual utility.

A particularly exciting question concerns the so-called *Price of Anarchy* [145, 190, 209]: How much better would the social welfare be if selfish players collaborated instead of seeking to maximize their own benefit? Technically, the Price of Anarchy is defined as the ratio between the social optimum and the worst Nash equilibrium.[1] In recent years, researchers have been fascinated to study the inherent loss of efficiency caused to a system by the participant's selfishness. Consequently, the Price of Anarchy [145, 208] and its complexity have been analyzed in various system settings, including the Internet [60, 83], wireless ad-hoc networks [71], or peer-to-peer systems [53]. Enforcing a truthful behavior or a reasonable efficiency in systems with a potentially high Price of Anarchy has been the goal of *algorithmic mechanism design*, e.g. [89, 187]. From a practical point of view, studying a system's Price of Anarchy indicates to what degree the system must be externally managed and controlled in order to maintain a satisfying global performance or outcome. Specifically, if a system has a large Price of Anarchy, it is necessary to design mechanisms (such as taxes, payment schemes, or coordination mechanisms) that force players to collaborate more efficiently. On the other hand, it may be feasible and reasonable to leave a system with a low Price of Anarchy to itself, because the selfish agents—by virtue of being selfish—are guaranteed to achieve an acceptable performance.

The first chapter of Part IV studies the Price of Anarchy and computational aspects of Nash equilibria in a specific peer-to-peer network setting, thereby attempting to shed light on the impact of selfish behavior in these networks. One of the key advantages of *structured P2P networks* as opposed to *unstructured P2P networks* is their potential to incorporate *locality-awareness* in their routing scheme. In this context, much theoretical research on structured P2P networks has focused on optimizing the *degree-stretch trade-off* at nodes and its efficient maintenance in view of churn [1, 5, 199, 210, 243]. Nonetheless, the majority of popular peer-to-peer systems currently used in reality are unstructured and each peer is (theoretically) capable of selecting its own set of peers in the system. In this case, what are the incentives to actually participate in a scheme?

Inspired by the work of Fabrikant et al. [83], we have studied what happens if nodes optimize their own locality by selecting preferable links. In more technical terms, we study a game theoretic network design problem in which peers are located in a metric space and every node can select directed links to an arbitrary subset of neighbors. Every node can select these nodes in such a way as to reduce its own costs, which are comprised of stretch-costs and maintenance-costs. We present tight bounds on the Price of Anarchy and also prove a complexity result on pure Nash equilibria of our game.

However, selfishness is not the only challenge to the performance of distributed systems. Frequently, systems have to cope with malicious *Byzantine adversaries* who seek to degrade the utility of the entire system, to

[1] A game may have many Nash equilibria of different social cost.

attack correctness of certain computations or to cause instability. In view of such threats, researchers—especially in the area of security and distributed computing—have devised ingenious solutions to defend against such possible attacks. But what is the impact of malicious players and Byzantine attacks on a system consisting of selfish players?

In Chapter 24, we extend current research on game theoretic aspects of networking by allowing some players to be *malicious* or *Byzantine* rather than selfish. The question is: What is the impact of Byzantine players on the system's efficiency compared to purely selfish environments or compared to the social optimum? In order to capture the efficiency degradation resulting from malicious Byzantine players, we introduce the notion of the *Price of Malice*. Technically, the Price of Malice is the ratio between the (regular) Nash equilibrium and a similarly defined so-called *Byzantine Nash equilibrium* which incorporates malicious players.

As a case study, we consider a simple virus-inoculation game, which models the containment of the spread of viruses. In this game, each node can choose whether or not to install anti-virus software. Then, a virus starts from a random node and iteratively infects all neighboring nodes which are not inoculated. Intriguingly, we obtain the result that depending on the amount of knowledge about the existence of malicious players in the system, the Price of Malice may actually become smaller than 1. That is, in certain settings, we can observe an actual *improvement* of overall system performance if malicious players interact with selfish players, as opposed to a system consisting of selfish players, only. This highlights a "fear-factor" that can also be observed in real life settings. Chapter 24 upper bounds this fear-factor in the virus inoculation game.

Before and during the time this thesis has been written, the field of algorithmic game theory and mechanism design has been among the most feverously studied topics in theoretical computer science. There have been numerous outstanding results in the recent past that have shed light on the fundamental nature of algorithmic game theory and its complexity, e.g. [59, 84, 191]. In contrast to such achievements, our contribution in this part of the thesis is of a smaller scale: We aim at providing insight into some specific game theoretic settings which are relevant in networking and distributed computing.

Chapter 23

The Locality Game: Topologies Formed by Selfish Peers

The power of peer-to-peer (P2P) computing arises from the collaboration of its numerous constituent parts, the peers. If all the participating peers contribute some of their resources—for instance bandwidth, memory, or CPU cycles—, highly scalable decentralized systems can be built which significantly outperform existing server based solutions. Unfortunately, in reality, many peers are selfish and strive for maximizing their own utility by benefiting from the system without contributing much themselves. Hence the performance—and thus its success in practice!—of a P2P system crucially depends on its capability of dealing with *selfishness*. A well-known mechanism designed to cope with this freeriding problem is the *tit-for-tat policy* which is for instance employed by the file-distribution tool *BitTorrent*.

However, selfish behavior in peer-to-peer networks may not be restricted to the peer's unwillingness to contribute bandwidth or memory. For example, in unstructured P2P systems—the predominant P2P architectures in today's Internet—, a peer can select to which and to how many other peers in the network it wants to connect. With a clever choice of neighbors, a peer can attempt to optimize its lookup performance by minimizing the latencies—or more precisely, the *stretch*—to the other peers in the network. Achieving good stretches by itself is of course simple: A peer can establish links to a large number of other peers in the system. Because the memory and maintenance overhead of such a neighbor set is large, however, egoistic peers try to keep stretches low, while avoiding to store too many neighbors. It is this trade-off between the need to have small latencies and the desire to reduce maintenance overhead that governs the decisions of selfish peers.

In order to analyze the impact of selfish neighbor selection on the quality of the resulting network topologies, we take a game theoretic approach. In

particular, we study the *Price of Anarchy* of P2P overlay creation, which is the ratio between an optimal solution obtained by perfectly collaborating participants compared to a solution generated by peers that act in an egoistic manner, optimizing their individual benefit. The Price of Anarchy in peer-to-peer systems quantifies the possible degradation caused by selfishness. Therefore, the Price of Anarchy is a measure that helps explaining the necessity (or non-necessity) of cooperation mechanisms in various aspects of these systems.

Interestingly, it turns out that the topologies of selfish, unstructured P2P systems can be much worse than in a scenario in which peers collaborate. More precisely, we show in Section 23.2 that the Price of Anarchy is $\Theta(\min(\alpha, n))$, where α is a parameter that captures the tradeoff between lookup performance (low stretches) and the cost of neighbor maintenance, and n is the number of peers in the system, respectively. Thereby, the upper bound $O(\min(\alpha, n))$ holds for peers located in *arbitrary* metric spaces, including the popular *growth-bounded* and *doubling metrics*. On the other hand, this bound is tight even in such a simple metric space as the 1-dimensional *Euclidean space*. As a second contribution, we prove in Section 23.3 that the topology of a static peer-to-peer system consisting of selfish peers may never converge to a stable state. That is, links may continuously change even in environments without *churn* (causing the network to be inherently instable). Furthermore, the problem of deciding whether a given peer-to-peer network is stable or not is NP-hard.

Related Work

The lack of cooperation in traditional P2P file-sharing systems has been well-documented over the last years [6, 239], and research on the causes and possible counter-measures is very active, e.g., [13] and [136]. Most of the current literature focuses on the issue of free resource consumption, *freeriding*.

Our game-theoretic model of *network creation* is inspired by the paper by Fabrikant et al. [83] which studies networks created by selfish agents. In particular, in the model of [83], agents correspond to nodes of a graph and every agent is free to build links to other nodes. The goal of each agent is to have a small (hop-)distance to all other nodes and to minimize its number of links. The network creation game of [83] has spurred a number of subsequent work in various settings, for instance [8, 14, 60, 71].

In contrast to these existing network creation games, our model takes into account many of the intrinsic properties of P2P systems. For instance, nodes are located in a *metric space* and the distances between nodes correspond to latencies. Our optimization function captures the *locality properties* of P2P systems, i.e., the desire to reduce latencies (expressed as the stretch) experienced when performing look-up operations. Finally, the fact that a peer can decide to which other peers it wishes to store pointers and thus maintain links yields a scenario with *directed links*.

Building *structured systems* that explicitly exploit locality properties has been a flourishing research area in networking and P2P computing (e.g. [1,

210, 236]). In early literature on distributed hash tables (DHT), the major measure of system quality has been the number of hops required for look-up operations. While this hop-distance is certainly of importance, it has been argued that the delay of communication (i.e., the stretch between pairs of peers) is a more relevant quality measure. Based on results achieved in [199], systems such as [1, 5, 210, 243] guarantee a provably bounded stretch with a limited number of links per peer. All of these systems are *structured* and peers are supposed to participate in a carefully predefined topology. In a sense, we complement this line of research by analyzing topologies as they are created by *selfish peers*, which are interested only in optimizing their *individual* trade-off between locality and maintenance overhead.

23.1 Model

Our model adapts the basic network creation game of Fabrikant et al in [83] in several ways. In particular, in our model, nodes are located in a metric space, links are directed, and each node's utility function includes the stretch to other nodes.

More specifically, we model the peers of a P2P network as points in a *metric space* $\mathcal{M} = (V, d)$, $V = \{v_0, v_1, \ldots, v_{n-1}\}$, where $d : V \times V \to [0, \infty)$ is the *distance function* which describes the underlying latencies between all pairs of peers. A peer $v_i \in V$ can choose to which subset of other peers it wants to store pointers (IP addresses). Formally, the *strategy space* of a peer v_i is given by $S_i = 2^{V \setminus \{v_i\}}$, and we refer to the actually chosen links as v_i's *strategy* $s_i \in S_i$. We say that v_i *maintains or establishes a link* to v_j if $v_j \in s_i$. The combination of all peers' strategies, i.e., $s = (s_0, ..., s_{n-1}) \in S_0 \times \cdots \times S_{n-1}$, yields a *directed* graph $G[s] = (V, \cup_{i=0}^{n-1}(\{v_i\} \times s_i))$, which describes the resulting P2P topology.

Selfish peers exploit *locality* in order to maximize their lookup performance. Concretely, a peer aims at minimizing the *stretch* to all other peers. As usual, the stretch between two peers v_i and v_j is the shortest distance between v_i and v_j using the links of the resulting topology G divided by the direct distance, i.e., for a topology G, $stretch_G(v_i, v_j) = d_G(v_i, v_j)/d(v_i, v_j)$. Clearly, it is desirable for a peer to have low stretches to other peers in order to keep its latency small. However, storing and especially maintaining a large number of links is expensive. For instance, the maintenance of a link may involve periodic pings to verify whether the neighbor is still alive. Therefore, the individual cost $c_i(s)$ incurred at a peer v_i is composed not only of the stretches to all other peers, but also of its *degree*, i.e., the number of its neighbors:

$$c_i(s) = \alpha \cdot |s_i| + \sum_{i \neq j} stretch_{G[s]}(v_i, v_j).$$

Note that this cost function captures the classic P2P trade-off between the need to minimize latencies and the desire to store and maintain only few links, as it has been addressed by many existing systems, for example Pastry [210]. Thereby, the relative importance of degree costs versus stretch

costs is expressed by the parameter α.

In order to evaluate the topologies constructed by selfish peers—and compare them with the topologies achieved by collaborating peers—, we study *Nash equilibria*. A topology constitutes a Nash equilibrium if no peer can reduce its individual cost by changing its set of neighbors given that the connections of all other peers remain the same. More formally, a (pure) Nash equilibrium is a combination of strategies s such that, for each peer v_i, and for all alternative strategies s' which differ only in the i^{th} component (different neighbor sets for peer v_i), $c_i(s) \leq c_i(s')$. This means that in a Nash equilibrium, no peer has an incentive to change its current set of neighbors, that is, Nash equilibria are *stable*.

While peers try to minimize their individual cost, the system designer is interested in a good overall quality of the P2P network. The *social cost* is the sum of all peers' individual costs, i.e.,

$$C(G,s) = \sum_i c_i(s) = \alpha|E| + \sum_{i \neq j} stretch_G(v_i, v_j).$$

The *Price of Anarchy* is the ratio between the social cost of the worst Nash equilibrium and the social cost of the optimal topology.

Determining the parameter α in real unstructured peer-to-peer networks is an interesting field for study. As mentioned, α measures the relative importance of low stretches compared to the peers' degrees, and thus depends on the system or application: For example, in systems with many lookups where good response times are crucial, α is smaller than in distributed archival storage systems consisting mainly of large files. In the sequel, we denote the link and stretch costs by $C_E(G) = \alpha|E|$ and $C_S(G) = \sum_{i \neq j} stretch_G(v_i, v_j)$, respectively.

23.2 Price of Anarchy

The Price of Anarchy is the ratio between the social cost of the worst Nash equilibrium and the social cost of the optimal topology. It is a measure that describes the degradation of a globally optimal solution caused by selfish individuals. In this section, we show that the topologies created by selfish peers deteriorate more (compared to collaborative networks) as the cost of maintaining links becomes more important (larger α). Concretely, in Section 23.2.1 we prove that for *arbitrary* metric spaces—thus, including the important and well-studied *growth-bounded* [139] and *doubling* (e.g. [40]) metrics—, the Price of Anarchy never exceeds $O(\min(\alpha, n))$. We then show in Section 23.2.2 that this bound is tight even in the "simplest" metric space, the 1-dimensional Euclidean space, where there exist Nash equilibria with a Price of Anarchy of $\Omega(\min(\alpha, n))$.

23.2.1 Upper Bound

Assume the most general setting where n peers are arbitrarily located in a given metric space $\mathcal{M}$, and consider a peer v_i which has to find a suitable

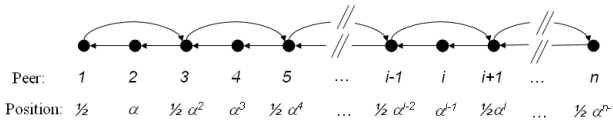

Figure 23.1: Example topology G where the Price of Anarchy is $\Theta(\min(\alpha, n))$ for $3.4 \leq \alpha$. The peers are arranged on a 1-dimensional Euclidean line, with exponentially increasing distances. Even peers are only connected to the nearest peer on the left, while odd peers additionally have a link to the second nearest peer on their right. Observe that every peer has stretch 1 to all peers on the left.

neighbor set. Clearly, the *maximal* stretch from v_i to any other peer v_j in the system is at most $\alpha + 1$: If $stretch(v_i, v_j) > \alpha + 1$, v_i could establish a direct link to v_j, reducing the stretch from more than $\alpha + 1$ to 1, while incurring a link cost of α. Therefore, in any Nash equilibrium, no stretch exceeds $\alpha + 1$. Because there are at most $n(n-1)$ directed links (from each peer to all remaining peers), the social cost of a Nash equilibrium is $O(\alpha n^2 + \alpha n^2)$. In the social optimum, on the other hand, all stretches are at least 1 and there must be at least $n - 1$ links in order to keep the topology connected. This lower bounds the social cost by $\Omega(\alpha n + n^2)$ and yields the following result.

Theorem 23.1. *For any metric space $\mathcal{M}$, the Price of Anarchy is in the order of $O(\min(\alpha, n))$.*

Theorem 23.1 implies that if the relative importance of the peers' stretch is large, the Price of Anarchy is small. That is, for small α, the selfish peers have an incentive to establish links to many other peers, while also the optimal network is highly connected.

23.2.2 Lower Bound

We now show that there are P2P networks in which the Price of Anarchy is as bad as $\Omega(\min(\alpha, n))$, which implies that the upper bound of Section 23.2.1 is asymptotically tight. Intriguingly, the Price of Anarchy can deteriorate to $\Theta(\min(\alpha, n))$ even if the underlying latency metric describes a simple 1-dimensional Euclidean space.

Consider the topology G in Figure 23.1 in which peers are located in a line, and the distance (latency) between two consecutive peers increases exponentially towards the right. Concretely, peer i is located at position $\alpha^{i-1}/2$ if i is odd, and at position α^{i-1} if i is even. The peers of G maintain links as follows: All peers have a link to their nearest neighbor on the left. Odd peers additionally have a link to the second nearest peer on their right. After proving that G constitutes a Nash equilibrium, we derive the lower bound on the Price of Anarchy by computing the social cost of this topology.

Lemma 23.2. *The topology G shown in Figure 23.1 forms a Nash equilibrium for $\alpha \geq 3.4$.*

Proof. We distinguish between even and odd peers. For both cases, we show that no peer has an incentive to deviate from its strategy.

Case even peers: Every even peer i needs to link to at least one peer on its left, otherwise i cannot reach the peers $j < i$. A connection to peer $i-1$ is optimal, as the stretch to all peers $j < i$ becomes 1. Observe that every alternative link to the left would imply a larger stretch to at least one peer on the left without reducing the stretch to peers on the right. Furthermore, i cannot reduce the distance to any—neither left nor right—peer by adding further links to the left. Hence, it only remains to show that i cannot benefit from adding more links to the right.

By adding a link to the right, peer i shortens the distance to *all* peers on the right. However, we show that the cost reduction per peer decreases as a geometric series, and any such link to the right would strictly increase i's costs. We consider two cases: i linking to an odd peer on the right, and i linking to an even peer on the right.

Link to an odd peer: Consider the benefit of i adding a link to its odd neighbor $i+1$. For an odd peer $j > i$, we define the *benefit* $B_{i,j}$ as the stretch cost reduction caused by the addition of the link $(i, i+1)$. We have, for $i \geq 2$,

$$\begin{aligned}
B_{i,j} &= stretch_{old}(i,j) - stretch_{new}(i,j) \\
&= \frac{d(i,i-1)+d(i-1,j)}{d(i,j)} - \frac{d(i,j)}{d(i,j)} \\
&= \frac{\alpha^{i-1} - \frac{1}{2}\alpha^{i-2} + \frac{1}{2}\alpha^{j-1} - \frac{1}{2}\alpha^{i-2}}{\frac{1}{2}\alpha^{j-1} - \alpha^{i-1}} - 1 \\
&= \frac{\frac{1}{2}\alpha^{j-1} + \alpha^{i-1} - \alpha^{i-2}}{\frac{1}{2}\alpha^{j-1} - \alpha^{i-1}} - \frac{\frac{1}{2}\alpha^{j-1} - \alpha^{i-1}}{\frac{1}{2}\alpha^{j-1} - \alpha^{i-1}} \\
&= \frac{2\alpha^{i-1} - \alpha^{i-2}}{\frac{1}{2}\alpha^{j-1} - \alpha^{i-1}} = \frac{2 - \frac{1}{\alpha}}{\frac{1}{2}\alpha^{j-i} - 1}
\end{aligned}$$

Similarly, the savings $B_{i,j}$ for an even peer $j > i$ and $i \geq 2$ amount to

$$\begin{aligned}
B_{i,j} &= stretch_{old}(i,j) - stretch_{new}(i,j) \\
&= \frac{d(i,i-1)+d(i-1,j+1)+d(j+1,j)}{d(i,j)} - \frac{d(i,j+1)+d(j+1,j)}{d(i,j)} \\
&= \frac{\alpha^{i-1} - \alpha^{i-2} + \alpha^{j} - \alpha^{j-1}}{\alpha^{j-1} - \alpha^{i-1}} - \frac{\alpha^{j} - \alpha^{i-1} - \alpha^{j-1}}{\alpha^{j-1} - \alpha^{i-1}} \\
&= \frac{2\alpha^{i-1} - \alpha^{i-2}}{\alpha^{j-1} - \alpha^{i-1}} = \frac{2 - \frac{1}{\alpha}}{\alpha^{j-i} - 1}
\end{aligned}$$

Hence, for all $\alpha \geq 3.4$, the total savings B_i for peer i are less than

$$\begin{aligned} B_i &= \sum_{\text{odd } j > i} B_{i,j} + \sum_{\text{even } j > i} B_{i,j} \\ &< \sum_{\delta=1}^{\infty} \frac{2-\frac{1}{\alpha}}{\frac{1}{2}\alpha^{2\delta-1}-1} + \sum_{\delta=1}^{\infty} \frac{2-\frac{1}{\alpha}}{\alpha^{2\delta}-1} \\ &\underset{(\alpha\geq 3)}{\leq} \left(2-\frac{1}{\alpha}\right) \sum_{\delta=1}^{\infty} \left(\frac{1}{\frac{1}{2}\alpha^{2\delta-2}} + \frac{1}{\alpha^{2\delta-1}} \right) \\ &= \left(2-\frac{1}{\alpha}\right)\left(\frac{2\alpha^2}{\alpha^2-1} + \frac{\alpha}{\alpha^2-1}\right) \\ &= \frac{4\alpha^2-1}{\alpha^2-1} \underset{(\alpha\geq 3.4)}{<} \alpha+1 \end{aligned}$$

Therefore, the construction of link $(i, i+1)$ would be of no avail (benefit smaller than cost). The benefit of alternative or additional links to odd neighbors on the right is even smaller.

Link to an even peer: A link to an even peer $j > i$ entails a stretch 1 to the corresponding peer instead of $stretch_{old}(i,j) = (\alpha^j - \alpha^{j-1} + \alpha^{i-1} - \alpha^{i-2})/(\alpha^{j-1} - \alpha^{i-1}) < \alpha+1$ for $\alpha > 2$. However, the stretch from i to all other peers remains unchanged, since the path $i \rightsquigarrow (i-1) \rightsquigarrow (i+1)$ is shorter than $i \rightsquigarrow (i+2) \rightsquigarrow (i+1)$: $\alpha^{i-1} - \frac{1}{2}\alpha^{i-2} + \frac{1}{2}\alpha^i - \frac{1}{2}\alpha^{i-2} < \alpha^{i+1} - \alpha^{i-1} + \alpha^{i+1} - \frac{1}{2}\alpha^i$ for $\alpha > 1$. Therefore, an even peer i has no incentive to build links to any even peer on its right.

Case odd peers: An odd peer i needs to link to peer $i-1$, otherwise there is no connection to $i-1$ and the stretch from i to $i-1$ is infinite. Moreover, if the link $(i, i-1)$ is established, $stretch(i,j) = 1$ for all $j < 1$. Therefore, peer i does not profit from building additional or alternative links to the left.

It remains to study links to the right. In order to reach all peers with a finite stretch, peer i needs a link to some peer $j \geq i+2$. In the following, we first show that peer i can always benefit from a link $(i, i+2)$, independently of additional links to the right. Secondly, we prove that if i has a link $(i, i+2)$, it has no incentive to add further links.

Assume peer i has no direct link to peer $i+2$. Then, $stretch(i, i+2) \geq (2\alpha^{i+2} - \frac{1}{2}\alpha^{i-1} - \frac{1}{2}\alpha^{i+1})/(\frac{1}{2}\alpha^{i+1} - \frac{1}{2}\alpha^{i-1}) > \alpha+1$. Hence, no matter which links it already has, peer i can benefit by additionally pointing to peer $i+2$. On the other hand, if i maintains the link $(i, i+2)$, any other links to the right only reduce i's gain. For *odd* peers, this is obvious, since the corresponding stretches are already optimal. A link (i,j) to some *even* peer $j > i$ only improves the stretch to peer j itself, but not to other peers. The stretch to peer j becomes 1 instead of $stretch_{old}(i,j) = (\frac{1}{2}\alpha^{j+1} - \frac{1}{2}\alpha^{i-1} + \frac{1}{2}\alpha^{j+1} - \alpha^j)/(\alpha^j - \frac{1}{2}\alpha^{i-1}) = (\alpha^{j+1} - \alpha^j - \frac{1}{2}\alpha^{i-1})/(\alpha^j - \frac{1}{2}\alpha^{i-1}) < \alpha+1$ for $\alpha > 0$. Thus, also this link would increase i's costs. □

Having verified that the topology of Figure 23.1 is a Nash equilibrium, we compute its social cost.

Lemma 23.3. *The social cost $C(G)$ of the topology G shown in Figure 23.1 is*

$$C(G) \in \Theta(\alpha n^2).$$

Proof. The topology G has $n-1$ links pointing to the left and $\lfloor n/2 \rfloor$ links pointing to the right. Hence, the total link costs are

$$C_E(G) = \alpha \left[(n-1) + \lfloor n/2 \rfloor \right] \in \Theta(\alpha n).$$

It remains to compute the costs of the stretches. The stretch from an odd peer i to an even peer $j > i$ is $stretch(i,j) = (\alpha^j - \alpha^{j-1} - \frac{1}{2}\alpha^{i-1})/(\alpha^{j-1} - \frac{1}{2}\alpha^{i-1}) > (\frac{1}{2}\alpha^j - \frac{1}{2}\alpha^{i-1})/(\alpha^{j-1} - \frac{1}{2}\alpha^{i-1}) > \frac{1}{2}\alpha$ for $\alpha > 2$. Thus, the sum of the stretches of an odd peer i is

$$\begin{aligned} C_S(i) &= \sum_{j<i} stretch(i,j) + \sum_{j>i} stretch(i,j) \\ &> (i-1) + \frac{1}{2}\alpha \left\lfloor (n-i-1)/2 \right\rfloor + \left\lfloor (n-i)/2 \right\rfloor . \end{aligned}$$

The stretch between two even peers i and j is $stretch(i,j) = (\alpha^j - \alpha^{j-1} + \alpha^{i-1} - \alpha^{i-2})/(\alpha^{j-1} - \alpha^{i-1}) > (\frac{1}{2}\alpha^j - \frac{1}{2}\alpha^{i-1})/(\alpha^{j-1} - \alpha^{i-1}) > \frac{1}{2}\alpha$ for $j > i$ and all $\alpha > 2$. Thus, the stretch costs are at least

$$C_S(i) > (i-1) + \frac{1}{2}\alpha \left(\left\lfloor (n-i-1)/2 \right\rfloor - 1 \right) + \left\lfloor (n-i-1)/2 \right\rfloor .$$

Adding up the stretches of odd and even peers yields a lower bound on the total stretch cost.

$$\begin{aligned} C_S(G) &= \sum_{i \text{ even}} C_S(i) + \sum_{i \text{ odd}} C_S(i) \\ &> \frac{n(n-2)}{2} + \frac{\alpha((n-3)(n-2)-n)}{8} + \frac{(n-1)(n-2)}{4} \\ &\in \Omega(\alpha n^2). \end{aligned}$$

Thus, in combination with Theorem 23.1, it follows that $C_S(G) \in \Theta(\alpha n^2)$. The proof is concluded by combining link and stretch costs,

$$C(G) = C_E(G) + C_S(G) \in \Theta(\alpha n^2).$$

□

Theorem 23.4. *The Price of Anarchy of the peer topology G shown in Figure 23.1 is* $\Theta(\min(\alpha, n))$.

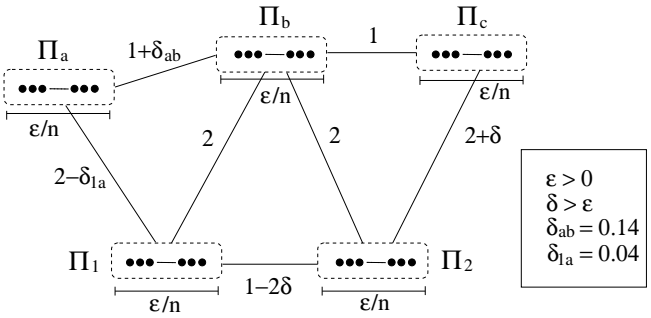

Figure 23.2: Instance I_k has no pure Nash equilibrium when $\alpha = 0.6k$, where $k = n/5$. The number of peers in each cluster is k.

Proof. The upper bound follows directly from the result obtained in Theorem 23.1. As for the lower bound, if $\alpha < 3.4$, the theorem holds because $O(\min(\alpha, n)) = O(1)$ in this case and the Price of Anarchy is always at least 1. By Lemma 23.2, the topology G constitutes a Nash equilibrium for $\alpha \geq 3.4$. Moreover, by Lemma 23.3, the social cost of G is in $\Theta\left(\alpha n^2\right)$. In the following, we prove that the optimal social cost is upper bounded by $O(n^2 + \alpha n)$ from which the claim of the theorem follows by dividing the two expressions.

Consider again the peer distribution shown in Figure 23.1, and assume that there are no links. If every peer connects to the nearest peer to its left and to the nearest peer to its right, there are $2(n-1)$ links, and all stretches are 1. Thus, the social cost of this resulting topology $\widetilde{G}$ is $C(\widetilde{G}) = \alpha \cdot 2(n-1) + n(n-1) \in O(n^2 + \alpha n)$. The optimal social cost is at most the social cost of $\widetilde{G}$. □

23.3 The Complexity of Nash Equilibrium

In the original network creation game studied in [83], there always exists a pure Nash equilibrium. Interestingly, this is not guaranteed in our locality game; there may not exist a pure Nash equilibrium for certain P2P networks. In other words, a system of selfish peers may never converge to a stable state, even in the absence of churn, mobility, or other sources of dynamics.

Theorem 23.5. *Regardless of the magnitude of α, there are metric spaces $\mathcal{M}$, for which there exists no pure Nash equilibrium, i.e. certain P2P networks cannot converge to a stable state. This is the case even if $\mathcal{M}$ is a 2-dimensional Euclidean space.*

Instead of presenting the formal proof (which will be implicit in the proof of Theorem 23.6), we attempt to highlight the main idea only. Assume

that the parameter α is a multiple of 0.6, i.e., $\alpha_k = 0.6k$ for an arbitrary integer $k > 0$. Given a specific k, the 2-dimensional Euclidean instance I_k of Figure 23.2 has no pure Nash equilibrium. Specifically, I_k constitutes a situation in which there are peers $v_1 \in \Pi_1$ and $v_2 \in \Pi_2$ that continue to deviate to a better strategy ad infinitum, i.e., the system cannot converge.

The n peers of instance I_k are grouped into five clusters Π_1, Π_2, Π_a, Π_b, and Π_c, each containing $k = n/5$ peers. Within a cluster, peers are located equidistantly in a line, and each cluster's diameter is ϵ/n, where $\epsilon > 0$ is an arbitrarily small constant. The *inter-cluster distance* $d(\Pi_i, \Pi_j)$ between Π_i and Π_j is the minimal distance between any two peers in the two clusters. Distances not explicitly defined in Figure 23.2 follow implicitly from the constraints imposed by the underlying Euclidean plane.

The proof unfolds in a series of lemmas that characterize the structure of the resulting topology $G[s]$ if the strategies s form a Nash equilibrium in I_k. First, it can be shown that in $G[s]$, two peers in the same cluster are always connected by a path that does not leave the cluster. Secondly, it can be shown that there exists exactly one link in both directions between clusters Π_a and Π_b, Π_b and Π_c, as well as between Π_1 and Π_2. A third structural characteristic of any Nash equilibrium is that for every i and j, there is *at most one* directed link from a cluster Π_i to peers in a cluster Π_j.

To preserve connectivity, some peers in Π_1 and Π_2 must have links to top-peers. Based on the aforementioned observations, the set of possible strategies can further be narrowed down as follows.

- Neither peers in Π_1 nor Π_2 select three links to top-peers.
- There exists a peer $v_1 \in \Pi_1$ that establishes a link to Π_a.
- There is exactly one link from cluster Π_2 to either cluster Π_b or Π_c, but there is no link to Π_a.

Correctness of all three properties is proven by verifying that there exists some node $v_1 \in \Pi_1$ or $v_2 \in \Pi_2$ that has an incentive to change its strategy in case the property is not satisfied. If, for instance, there are two peers $v_2, v_2' \in \Pi_2$ that simultaneously maintain links to Π_b and Π_c, (e.g. v_2 to Π_b and v_2' to Π_c, thus violating case iii)), v_2' can lower its costs by dropping its link to Π_c. Intuitively, this holds because the sum of the stretches $\sum_{v_c \in \Pi_c} stretch(v_2', v_c)$ entailed by the indirection $v_2' \rightsquigarrow v_2 \rightsquigarrow \Pi_b \rightsquigarrow \Pi_c$ does not justify the additional cost α.

It can be shown that only the six structures depicted in Figure 23.3 remain valid candidates for Nash topologies. In each scenario, however, at least one peer benefits from deviating from its current strategy.

Case 1: In this case, a peer $v_1 \in \Pi_1$ can reduce its cost by adding a link to a peer in Π_b.

Case 2: If the only outgoing link from Π_1 to a top-cluster is to cluster Π_a, the peer $v_2 \in \Pi_2$ maintaining the link to Π_c can be shown to profit from switching its link from Π_c to Π_b.

Case 3: The availability of the link from Π_1 to Π_b changes the optimal

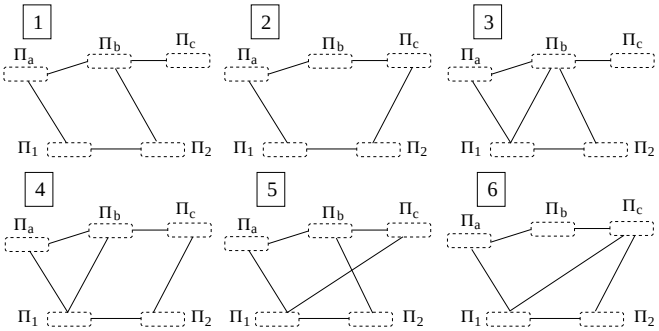

Figure 23.3: Candidates for a Nash equilibrium.

choice of the above mentioned peer $v_2 \in \Pi_2$. Unlike in the previous case, v_2 now prefers linking to Π_c instead of Π_b.

Case 4: Due to the existence of a link from a peer $v_2 \in \Pi_2$ to Π_c, the peer $v_1 \in \Pi_1$ with the link to Π_b has an incentive to drop this link and instead use the detours via Π_2 and Π_a to connect to Π_c and Π_b, respectively.

Case 5: In this case, the peer $v_1 \in \Pi_1$ having the link to Π_c reduces its cost by replacing this link with a link to a peer in Π_b.

Case 6: Finally, this case is similar to Case 4 in the sense that $v_1 \in \Pi_1$ with the link to Π_b has an incentive to remove its link to Π_c

These cases highlight how the system is ultimately trapped in an infinite loop of strategy changes, without ever converging to a stable situation. There is always at least one peer which can reduce its cost by changing its strategy. For instance, the following sequence of topology changes could repeat forever (cf. Figure 23.3): $1 \rightsquigarrow 3 \rightsquigarrow 4 \rightsquigarrow 2 \rightsquigarrow 1 \rightsquigarrow 3 \ldots$ In other words, selfish peers will not achieve a stable network topology.

The question is whether for a given P2P network, it can be determined if it will eventually converge to a stable state or not. In the following, we show that it is NP-hard to decide whether there exists a pure Nash equilibrium. This result establishes the *complexity of stability* in unstructured P2P networks, showing that in general, it is impossible to determine whether a peer-to-peer network consisting of selfish peers can stabilize or not.

Ever since Papadimitriou's influential survey on game theoretic aspects of the Internet [190], the complexity of Nash equilibria has become one of the most active and fruitful areas in recent theoretical computer science research. Numerous profound results on important families of games have been presented for instance in [59, 84, 191]. In the sequel, we aim at a much more humble result, namely, we want to show that it is NP-hard to decide whether our locality game in a given P2P network has a pure Nash equilibrium.

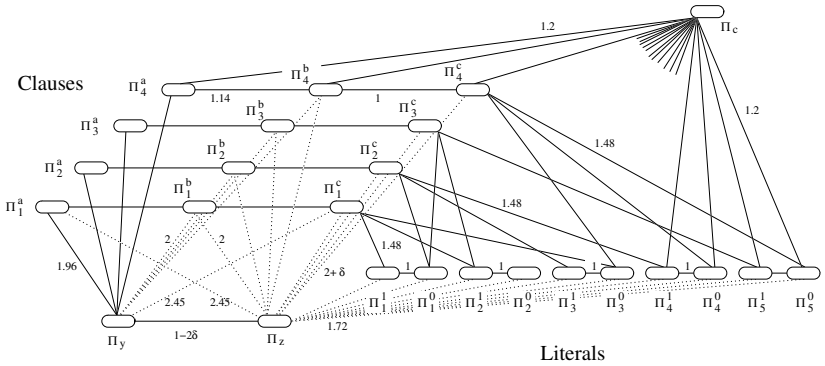

Figure 23.4: The graph G_I for instance $I = (x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_3 \vee x_4) \wedge (\overline{x_1} \vee x_2 \vee x_5) \wedge (\overline{x_3} \vee \overline{x_4} \vee \overline{x_5})$. Each cluster contains k peers with pairwise distance of ϵ. The constant δ is an arbitrarily constant such that $\delta > \epsilon > 0$.

Theorem 23.6. *Regardless of the magnitude of α, determining whether a given P2P network represented by a metric space $\mathcal{M}$ has a pure Nash equilibrium (and can therefore stabilize) is NP-hard.*

The proof being rather technical, we begin by briefly describing its main intuition. The proof is based on a reduction from an NP-complete special form of 3-SAT in which each variable appears in at most 3 clauses [102]. For any α a multiple of 0.6, i.e., $\alpha_k = 0.6k$ for an arbitrary integer $k > 0$, we give a polynomial time construction of a metric space $\mathcal{M}_I^k$ from an instance I of 3-SAT, such that the following holds: There exists a pure Nash equilibrium in $\mathcal{M}_I^k$ if and only if I is satisfiable.

The reduction is illustrated in Figure 23.4, each rectangular box representing a cluster of k peers. Assume that the 3-SAT instance is given in standard CNF normal form. For each clause C_j, we employ a gadget of three *clause-clusters* Π_j^a, Π_j^b, and Π_j^c. For every variable x_i, the two *literal-clusters* Π_i^0 and Π_i^1 represent the negative and positive literal of the variable, respectively. Finally, the construction's peer set is completed with three special clusters Π_c, Π_y, and Π_z. The pairwise distances between two peers in $\mathcal{M}_I^k$ are determined by the undirected weighted graph G_I^k shown in Figure 23.4. Two nodes within the same cluster have a distance of ϵ, for some arbitrarily small $\epsilon < (k(2n + 3m + 3))^{-2}$, where m and n denote the number of clauses and variables in I, respectively. An edge of G_I^k describes the *cluster-distance* between two clusters. That is, the mutual distance between every pair of two peers $v_i \in \Pi_i$ and $v_j \in \Pi_j$ in neighboring clusters Π_i and Π_j with cluster-distance X is $d(v_i, v_j) = X$. All other distances are determined by the length of the shortest path between the peers in G_I^k, i.e., $\mathcal{M}_I^k$ corresponds to the *shortest path metric* induced by G_I^k. Note that while $\mathcal{M}_I^k$ cannot be embed-

ded in the Euclidean space, it still forms a valid metric space, i.e., it fulfils symmetry, triangle inequality, and no two peers have the same location.

Consider an arbitrary clause C_j. Its clause-clusters Π_j^a, Π_j^b, and Π_j^c in combination with the two special clusters Π_y and Π_z form an instance similar to I_k as used in the discussion of Theorem 23.5 (cf Figure 23.2). Hence, intuitively, when considering such a clause-gadget by itself, it does not have a pure Nash equilibrium. In order to make a clause-gadget stable, however, literal clusters may be used. For this purpose, the cluster-distance between each pair of corresponding literals is 1 and peers in Π_z have a distance of 1.72 to all literal-peers. Furthermore, the distance between a clause-cluster Π_j^c and a literal-cluster depends on whether the corresponding literal appears in the clause. Specifically, if the positive literal x_i appears in clause C_j, $x_i \in C_j$, the distance between Π_i^1 and Π_j^c is small, i.e., only 1.48. Similarly, if $\overline{x_i} \in C_j$, then $d(\Pi_i^1, \Pi_j^c) = 1.48$. And finally, if neither literal is in C_j, then there exists no short connection between the clusters, and the shortest distance between peers in these clusters is via the central cluster Π_c.

The proof comprises two ingredients. First, we prove that if the underlying SAT instance I is *not satisfiable*, then there exists no Nash equilibrium. Towards this end, we show that in any Nash equilibrium two "neighboring" clusters (clusters connected by a short link in G_I^k, such as two clause-clusters in the same clause, a literal-cluster Π_i^1 to a clause-cluster Π_j^c if $x_i \in C_j$, or Π_c to all clause-clusters and literal-clusters,...) always establish links in both directions between them. Specifically, it can be shown that between such close-by clusters, there are always exactly two links, one in each direction. Furthermore, for every variable x_i, there is exactly one peer $v_z \in \Pi_z$ that establishes a link to exactly either Π_i^1 or Π_i^0 (but not both!), while no other peer in Π_z links to these clusters.

From these lemmas, it then follows that because I is not satisfiable, there must exist a clause C_{j^*} for which the path from $v_z \in \Pi_z$ to peers in $\Pi_{j^*}^c$ via any literal-peer has length at least $d(\Pi_z, \Pi_i^\mu) + d(\Pi_i^\mu, \Pi_i^{1-\mu}) + d(\Pi_i^{1-\mu}, \Pi_{j^*}^c) = 4.2$, for $\mu \in \{0, 1\}$. This path being long, it follows that it is worthwhile for v_z to build an additional link directly to some peer in $\Pi_{j^*}^c$ or even in $\Pi_{j^*}^b$ instead. Based on these observations, we show that the subset of $\mathcal{M}_I^k$ induced by peers in Π_y, Π_z, and the clause-peers of C_{j^*} behaves similarly as in instance I_k of Figure 23.2. That is, peers in Π_y and Π_z continue to change their respective strategies forever, thus preventing the system from stabilizing.

On the other hand, if the SAT instance I has a *satisfying assignment* A_I, we explicitly construct a set of pure strategies that constitute a Nash equilibrium. In this strategy vector, one peer in Π_z builds a direct link to a peer in Π_i^1 if x_i is set to true in A_I and to a peer in Π_i^0 otherwise. Since A_I is a satisfying assignment, there must exist a path from Π_z via a single literal-cluster (i.e., without the additional detour of going from one literal-cluster to the other) to peers in every cluster Π_j^c. This path can be shown to have length at most $k\epsilon + d(\Pi_z, \Pi_i^\mu) + k\epsilon + d(\Pi_i^\mu, \Pi_j^c) + k\epsilon = 3.2 + 3k\epsilon$ from Π_z via a literal-cluster to peers in every cluster Π_j^c. It follows that in any satisfied clause C_j, the achievable reduction in stretch costs at a peer in Π_z when connecting directly to clusters Π_j^b or Π_j^c is significantly smaller than in

an unsatisfied clause. Specifically, it can be shown that peers in Π_y and Π_z are in a *stable* situation if one peer $v_y \in \Pi_y$ connects to Π_j^a and Π_j^b of every clause C_j, and no peer in Π_z directly builds a link to any clause-peer. Since A_I is a satisfying assignment, peers in Π_y and Π_z are stable relative to *all* clauses in the SAT instance.

Furthermore, we also prove that in our strategy vector, no other peer in the network (i.e., peers in Π_c, Π_j^a, Π_j^b, Π_j^c, Π_i^1, or Π_i^0) has an incentive to deviate from its strategy. For this final ingredient of the proof, the existence of cluster Π_c is essential, because it ensures that the various helper peers are mutually connected by optimal paths.

Therefore, the P2P network induced by the metric space $\mathcal{M}_I^k$ has a pure Nash equilibrium if and only if the underlying SAT instance I is satisfiable. And hence, determining whether a given P2P network can ever stabilize is NP-hard. In the sequel, we turn this intuition into a formal proof. We begin in Section 23.3.1 by giving the construction of G_I^k (and consequently its shortest path metric $\mathcal{M}_I^k$) from the 3-SAT instance I. In Sections 23.3.2 and 23.3.3, we show that there exists a Nash equilibrium in $\mathcal{M}_I^k$ if and only if I is satisfiable. Theorem 23.6 then follows from Lemmas 23.16 and 23.18, as well as the NP-hardness of 3-SAT.

23.3.1 The Construction of $\mathcal{M}_\mathbf{I}^\mathbf{k}$

Let I be an instance of 3-SAT expressed in conjunctive normal form (CNF), in which each clause contains 3 literals. Without loss of generality, we can assume that each variable in I appears in at most 3 clauses [102]. Furthermore, we can restrict our attention to those instances of 3-SAT in which every variable appears in most 2 positive and 2 negative literals, because otherwise, the variable appears as a positive or negative literal only, which renders assigning a feasible value to this variable trivial. The set of clauses and variables of I is denoted by $\mathcal{C}$ and $\mathcal{X}$, respectively. Further, we write $m = |\mathcal{C}|$ and $n = |\mathcal{X}|$. Given I, we construct an undirected weighted graph $G_I^k = (V_I, E_I)$ in which each node represents a peer of the underlying network. Nodes are grouped into clusters of k peers and each cluster is illustrated as a rectangular box in Figure 23.4. Within each cluster, the pairwise distance between two peers is $\epsilon < (k(2n + 3m + 3))^{-2}$, and the distance between two peers in neighboring clusters is described by the *cluster-distance* $d(\Pi_i, \Pi_j)$ illustrated in Figure 23.4. The P2P network is then characterized by $\mathcal{M}_I^k$, which is induced by the *shortest path metric* of G_I^k, i.e., the distance between two peers corresponds to the length of the shortest path in G_I^k.

In more detail, G_I^k is defined as follows. The node-set V_I consists of three clusters of peers per clause $C_j \in \mathcal{C}$, denoted as *clause-clusters* Π_j^a, Π_j^b, and Π_j^c. Also, we add a pair of *literal-clusters* Π_i^0 and Π_i^1 for each of the n variables, with Π_i^0 representing the positive literal x_i, and Π_i^1 representing the negative literal $\overline{x_i}$. The set of clause-peers and literal-peers is denoted by C_P and L_P, respectively. Finally, there are three additional special clusters Π_c, Π_x, and Π_y. Call the union of Π_c and all clusters in C_P and L_P *top-layer clusters*. Peers in top-layer clusters are *top-layer peers*. The total number of peers N

in the network $\mathcal{M}_I^k$ is therefore $N = k(2n + 3m + 3)$. Notice that $N \cdot \epsilon$ is smaller than $(k(2n + 3m + 3))^{-1}$.

The pairwise distances between the peers in different clusters—as illustrated in Figure 23.4—are formally defined as

$$
\begin{array}{rll}
 & \forall v_c \in \Pi_c, \forall v_w \in C_P \cup L_P : & d(v_c, v_w) = 1.2 \\
\forall C_j \in \mathcal{C} : & \forall v_y \in \Pi_y, \forall v_j^a \in \Pi_j^a : & d(v_y, v_j^a) = 1.96 \\
\forall C_j \in \mathcal{C} : & \forall v_y \in \Pi_y, \forall v_j^b \in \Pi_j^b : & d(v_y, v_j^b) = 2 \\
\forall C_j \in \mathcal{C} : & \forall v_y \in \Pi_y, \forall v_j^c \in \Pi_j^c : & d(v_y, v_j^c) = 2.45 \\
\forall C_j \in \mathcal{C} : & \forall v_z \in \Pi_z, \forall v_j^a \in \Pi_j^a : & d(v_z, v_j^a) = 2.45 \\
\forall C_j \in \mathcal{C} : & \forall v_z \in \Pi_z, \forall v_j^b \in \Pi_j^b : & d(v_z, v_j^b) = 2 \\
\forall C_j \in \mathcal{C} : & \forall v_z \in \Pi_z, \forall v_j^c \in \Pi_j^c : & d(v_z, v_j^c) = 2 + \delta \\
\forall C_j \in \mathcal{C} : & \forall v_j^a \in \Pi_j^a, \forall v_j^b \in \Pi_j^b : & d(v_j^a, v_j^b) = 1.14 \\
\forall C_j \in \mathcal{C} : & \forall v_j^b \in \Pi_j^b, \forall v_j^c \in \Pi_j^c : & d(v_j^b, v_j^c) = 1 \\
\forall x_i \in \mathcal{X} : & \forall v_i^0 \in \Pi_i^0, \forall v_i^1 \in \Pi_i^1 : & d(v_i^0, v_i^1) = 1 \\
\forall x_i \in \mathcal{X} : & \forall v_z \in \Pi_z, \forall v_i^\mu \in \Pi_i^\mu : & d(v_z, v_i^\mu) = 1.72 \\
\forall C_j \in \mathcal{C}, x_i \in C_j : & \forall v_i^1 \in \Pi_i^1, \forall v_j^c \in \Pi_j^c : & d(v_i^1, v_j^c) = 1.48 \\
\forall C_j \in \mathcal{C}, \overline{x_i} \in C_j : & \forall v_i^0 \in \Pi_i^0, \forall v_j^c \in \Pi_j^c : & d(v_i^0, v_j^c) = 1.48 \\
 & \forall v_y \in \Pi_y, \forall v_z \in \Pi_z : & d(v_y, v_z) = 1 - 2\delta
\end{array}
$$

for δ being an arbitrarily small constant with $\delta > 10k\epsilon$, and $\mu \in \{0, 1\}$. All other distances not explicitly defined follow from the shortest path metric induced by the above definitions.

Intuitively, the idea of the construction is the following. Each clause $C_j \in \mathcal{C}$ is represented by a gadget consisting of the two clusters Π_y, Π_z, as well as the clause-clusters Π_j^a, Π_j^b, and Π_j^c. By itself, each such gadget is reminiscent of the construction shown in Figure 23.2. Specifically, this implies that the sub-network induced by each such clause-gadget does not have a pure Nash equilibrium when considered independently from the rest of the network.

In order to render a clause-gadget stable, literal-peers can be used. In particular, it can be shown that for $\mu \in \{0, 1\}$, the peers in every literal-cluster Π_i^μ construct links to those (at most two) clause-clusters Π_j^c in whose clause the literal occurs, that is, if $x_i^\mu \in C_j$. Based on this and other structural properties of Nash equilibria in $\mathcal{M}_I^k$, it can further be shown that in a Nash equilibrium, there is exactly one link from cluster Π_z to each variable $x_i \in \mathcal{X}$, i.e., one peer in Π_z connects to a peer in either Π_i^0 or Π_i^1 for all $x_i \in \mathcal{X}$.

Consider a clause C_j. If there is a peer $v_z \in \Pi_z$ that connects to at least one literal-cluster that is directly connected to Π_j^c, the length of the path from v_z to peers in Π_j^c via this literal-cluster is at most $k\epsilon + d(\Pi_z, \Pi_i^\mu) + k\epsilon + d(\Pi_i^\mu, \Pi_j^c) + k\epsilon = 3.2 + 3k\epsilon$. In this case, the detour from v_z to Π_j^c via some "satisfying" literal-cluster Π_i^μ—while being suboptimal compared

to the direct connection—is relatively small. Specifically, it is small enough to ensure that no peer in Π_z has an incentive to construct an additional direct link to Π_j^b or Π_j^c. Once peers in Π_z have no further need to establish direct links to a clause-peer of C_j, the best possible strategy of peers in Π_y becomes fixed, too. In other words, this satisfying literal helps in *stabilizing* the clause-gadget.

Conversely, if there is a clause C_j for which no peer in Π_z connects to a satisfying literal-cluster, there exists no efficient detour. Specifically, the length of the path from $v_z \in \Pi_z$ to $v_j^c \in \Pi_j^c$ via a literal-cluster is at least 4.2, including the distance between the positive and negative literal-cluster of the variable. The increased length of the detour renders the resulting stretch from Π_z to Π_j^c too large, and it becomes worthwhile for $v_z \in \Pi_z$ to construct direct links to Π_j^c, and even to Π_j^b. That is, in a sense, the network induced by the unsatisfied clause C_j becomes independent of the remainder of the network and therefore does not stabilize.

Finally, the special cluster Π_c ensures that the shortest path in G_I^k (and hence the distance in $\mathcal{M}_I^k$) between two top-layer peers is small. In fact, it can be shown that there are links in both directions from every top-layer cluster to Π_c. This implies that all top-layer clusters are connected to one another almost optimally in every Nash equilibrium, thus facilitating the proof that such an equilibrium exists in case I is satisfiable. We end the section with a series of lemmas that capture important structural properties of $\mathcal{M}_I^k$.

Lemma 23.7. *Consider two peers v_g and v_g' in an arbitrary cluster Π_g. In a Nash equilibrium, there exists a path from v_g to v_g' of length at most $k\epsilon$.*

Proof. Because the distance between v_g and v_g' is ϵ, it is easy to see that the shortest path between these two nodes must be located entirely in Π_g. Because the distance between each pair of peers in a cluster is ϵ and there are k peers in the cluster, the claim follows. □

Lemma 23.8. *Consider two arbitrary clusters Π_g and Π_h. In a Nash equilibrium, there is at most one peer $v_g \in \Pi_g$ that has a link to a peer in Π_h.*

Proof. Assume for contradiction that there are two nodes v_g and v_g' that maintain links to peers in Π_h. Then, v_g' can reduce its cost by dropping its link. Doing so, the stretches to each node in the network can increase by at most $2k\epsilon$. By the definition of ϵ, it holds that $2Nk\epsilon < \alpha$ and hence, dropping the link is worthwhile. □

Based on these two lemmas, we can go on to prove more elaborate properties.

Lemma 23.9. *Let Π_g and Π_h be two clusters with cluster distance at most $d(\Pi_g, \Pi_h) \leq 1.48$. In any Nash equilibrium, there is exactly one peer $v_g \in \Pi_g$ that has a link to a peer in Π_h.*

Proof. By Lemma 23.8, there cannot be more than one peer in Π_g that has a link to Π_h. It therefore remains to show that at least one link exists. We divide the proof in two parts and begin by showing that the claim holds for all pairs of clusters with cluster distance $d(\Pi_g, \Pi_h) \leq 1.2$. In a second step, we prove the claim for pairs of clusters with cluster distance $d(\Pi_g, \Pi_h) = 1.48$, which suffices because there are no cluster distances between 1.2 and 1.48 in G_I^k.

Consider any two clusters in the network $\mathcal{M}_I^k$ with cluster distance at most 1.2. It follows from the construction of G_I^k that the shortest path between peers in these clusters via a third cluster has length at least 2.2 (e.g., from Π_i^0 via Π_i^1 to Π_c). In other words, if there is no direct connection between the two clusters, v_g has a stretch of at least 2.2/1.2 to each peer in Π_h. Because $\frac{2.2k}{1.2} > \alpha + k(1 + 2k\epsilon)$, it is beneficial for v_g to establish a direct link to the other cluster.

For the second part of the proof, consider pairs of clusters with cluster distance $d(\Pi_g, \Pi_h) = 1.48$. Specifically, we need to show the existence of a link in each direction between clusters Π_j^c and Π_i^1, if $x_i \in C_j$, or between Π_j^c and Π_i^0, if $\overline{x_i} \in C_j$. The shortest indirect connection between two such clusters has length at least 2.4 (via cluster Π_c) and hence, the cumulated stretch to all peers in the respective cluster without a direct link is $\frac{2.4k}{1.48} > \alpha + k(1 + 2k\epsilon)$. That is, peers in both clusters decrease their cost by paying for this direct link. □

Notice that Lemma 23.9 implies that within a clause, neighboring clause-clusters (i.e., $\Pi_j^a \leftrightarrow \Pi_j^b$ and $\Pi_j^b \leftrightarrow \Pi_j^c$, respectively) are connected in both directions in any Nash equilibrium. The same holds for corresponding literal-cluster Π_i^1 and Π_i^0, as well as for a literal-cluster Π_i^1 (or Π_i^0) and a Π_j^c if $x_i \in C_j$ (or $\overline{x_i} \in C_j$). Also, there are links in both directions from any top-layer (clause or literal) cluster to Π_c and vice versa. All in all, this implies that in a Nash equilibrium, every pair of top-layer peers is connected almost optimally, i.e., with stretch of less than $1 + 2k\epsilon$. The value ϵ being defined to be smaller than $(k(m + n + 3))^{-2}$, this stretch is virtually as good as 1. Finally, there are also links from Π_y to Π_z and vice versa in any Nash equilibrium. In the sequel of the proof, we often implicitly use the fact that these "short" links are available in any Nash equilibrium without particular mention.

Lemma 23.10. *In a Nash equilibrium, there is exactly one peer $v_y \in \Pi_y$ that has a link to a peer in Π_j^a, for all $C_j \in \mathcal{C}$, and vice versa.*

Proof. Consider a specific Π_j^a. If there exists no direct link from Π_y to Π_j^a, the stretch of a peer $v_y \in \Pi_y$ to each peer in Π_j^a is at least $\frac{3.14}{1.96}$. Because for small enough ϵ, we have $\frac{3.14k}{1.96} > \alpha + k(1 + 2k\epsilon)$, it is always worthwhile for some v_y to build an additional link to Π_j^a. Clearly, the argument also holds for the opposite direction. □

Lemma 23.11. *Assume that there is a link between Π_z and at least one literal-cluster of every variable $x_i \in \mathcal{X}$ and that there is a link between Π_y*

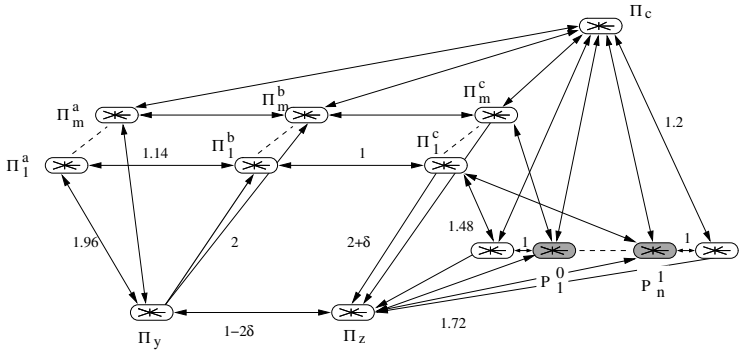

Figure 23.5: An example instance G_I^k with the topology resulting from strategy s. Within each cluster, the peers are connected as a star. Directed arrows between clusters indicate inter-cluster links between cluster-leaders. Cluster-leader $\hat{v}_z$ connects to those leaders of literal-peers that appear in the satisfying assignment A_I. In the example, A_I sets $x_1 = 0$ and $x_n = 1$.

and Π_j^a, for all $C_j \in \mathcal{C}$. Assume further that there are links in both directions between clusters with cluster distance at most 1.48. Finally, assume that all peers are connected within their cluster with a path of length at most $k\epsilon$. It holds for all j that the shortest path from a peer $v_y \in \Pi_y$ to a peer in $V \setminus (\Pi_j^a \cup \Pi_j^b \cup \Pi_j^c)$ is not via Π_j^a, Π_j^b, or Π_j^c, even when directly connecting to such a cluster. The same holds for $v_z \in \Pi_z$.

Proof. Recall that by assumption there exists a link from Π_y to Π_j^a (for every $C_j \in \mathcal{C}$) and Π_z. Hence, connecting to Π_j^b or Π_j^c clearly cannot reduce the stretch to peers in Π_z, Π_c, and any $\Pi_{j'}^a$, $j \neq j'$. Furthermore, the distance in the topology to any clause-peer in $\Pi_{j'}^b$ and $\Pi_{j'}^c$ via $\Pi_{j'}^a$ is at most $3.1 + 3k\epsilon$ and $4.1 + 4k\epsilon$, respectively, which is strictly smaller than $2 + 2 \cdot 1.2 = 4.4$, which is the shortest achievable distance via Π_j^b or Π_j^c. Finally, the path from $v_y \in \Pi_y$ to any literal-peer in Π_i^μ has length at most $3.72 - 2\delta + 3k\epsilon$. This is because there exists a link between Π_y and Π_z, and between Π_i^0 and Π_i^1, and because there is a link from Π_z to either Π_i^0 or Π_i^1. On the other hand, the path from $v_y \in \Pi_y$ to a literal-peer via Π_j^b or Π_j^c has length at least $2.45 + 1.48 = 3.93$. Similar arguments show that the same holds for $v_z \in \Pi_z$. □

23.3.2 Satisfiable Instances

In this section, we show that if I has a satisfying assignment A_I, then there exists a Nash equilibrium in $\mathcal{M}_I^k$. For this purpose, we explicitly construct a

set of strategies s, which we prove to constitute a Nash equilibrium. As for notation, we define $A_I(x_i)$ to be the assignment of x_i in A_I, i.e.,

$$A_I(x_i) := \begin{cases} 1 \ , \ x_i \text{ is set to 1 in } A_I \\ 0 \ , \ x_i \text{ is set to 0 in } A_I. \end{cases} \tag{23.1}$$

Furthermore, we define in every cluster Π_g a single *leader peer*, which we denote by $\hat{v}_g$. The role of this leader-peer is to construct all inter-cluster links going from this cluster to peers located in other clusters. The strategy of the remaining *non-leader peers* $\check{v}_g \in \Pi_g \setminus \{\hat{v}_g\}$ is to connect to the unique leader peer within their cluster. Formally, the strategy s_g for a non-leader peer $\check{v}_g \in \Pi_g \setminus \{\hat{v}_g\}$ is

$$s_g := \{\hat{v}_g\}.$$

For each leader-peer, we define the set of strategies s as follows:

$$\begin{aligned}
s_y &:= \Pi_y \cup \{\hat{v}_z\} \cup \bigcup_{C_j \in \mathcal{C}} \{\hat{v}_j^a, \hat{v}_j^b\} \\
s_z &:= \Pi_z \cup \{\hat{v}_y\} \cup \bigcup_{x_i \in \mathcal{X}} \{\hat{v}_i^{A_I(x_i)}\} \\
s_c &:= \Pi_c \cup \bigcup_{x_i \in \mathcal{X}} \{\hat{v}_i^0 \cup \hat{v}_i^1\} \cup \bigcup_{C_j \in \mathcal{C}} \{\hat{v}_j^a \cup \hat{v}_j^b \cup \hat{v}_j^c\} \\
s_j^a &:= \Pi_j^a \cup \{\hat{v}_c, v_y, \hat{v}_j^b\} && ,\forall C_j \in \mathcal{C} \\
s_j^b &:= \Pi_j^b \cup \{\hat{v}_c, \hat{v}_j^a, \hat{v}_j^c\} && ,\forall C_j \in \mathcal{C} \\
s_j^c &:= \Pi_j^c \cup \{\hat{v}_c, \hat{v}_z, \hat{v}_j^b\} \cup \bigcup_{x_i^\mu \in C_j} \{\hat{v}_i^\mu\} && ,\forall C_j \in \mathcal{C} \\
s_i^\mu &:= \Pi_i^\mu \cup \{\hat{v}_c, \hat{v}_z, \hat{v}_i^{1-\mu}\} \cup \bigcup_{x_i^\mu \in C_j} \{\hat{v}_j^c\} && ,\forall x_i \in \mathcal{X}
\end{aligned}$$

Strategy s is illustrated in Figure 5. Our goal is to show that s constitutes a Nash equilibrium. The topology resulting from strategy s contains all "short" links, i.e., links between cluster leaders of clusters that have a distance of at most 1.48 (cf Lemma 23.9). Additionally, peer $\hat{v}_y$ builds links to clause-cluster leaders $\hat{v}_j^a$ and $\hat{v}_j^b$ for all $C_j \in \mathcal{C}$. On the other hand, leaders $\hat{v}_j^a$ and $\hat{v}_j^c$ have a link to $\hat{v}_y$ and $\hat{v}_z$, respectively. Most importantly, however, for every variable $x_i \in \mathcal{X}$, leader-peer $\hat{v}_z$ maintains a link to the literal-peers $\hat{v}_i^{A_I(x_i)}$ that are used in the satisfying assignment. Note that because in s, peer $\hat{v}_z$ has exactly one connection to a literal-peer of every variable, we can apply Lemma 23.11. That is, no peer in clusters Π_y and Π_z can reduce its stretch to any peer $V \setminus (\Pi_j^a \cup \Pi_j^b \cup \Pi_j^c)$ by connecting to one of the clause-peers of clause C_j. Finally, note that non-leaders are directly connected to their cluster leader, and cluster leaders maintain direct links to each peer in their cluster.

The next three lemmas prove that no node has an incentive to single-handedly deviate from strategy s. In the proofs, we use the notation $\Delta_i(\psi)$ to denote the change in cost when peer v_i changes its strategy according to action ψ, ψ being clear from the context. Specifically, if $\Delta_i(\psi) \geq 0$, peer v_i has no incentive to perform action ψ because doing so would increase its cost.

We begin with a lemma that shows that no peer can unilaterally benefit from changing its links within its own cluster.

Lemma 23.12. *In s, no peer in an arbitrary cluster Π_g has an incentive to change its strategy within the cluster, i.e., to add, replace, or remove links to peers in Π_g.*

Proof. The cluster leader $\hat{v}_g$ cannot remove any link because the topology would become disconnected without it. Next, consider a non-leader $\check{v}_g$. If $\check{v}_g$ removes its link to the cluster-leader, it disconnects itself from the network. Adding one or more new link to a non-leader costs α per link, while the resulting stretch reduction per link is $\frac{2\epsilon}{\epsilon} - 1 = 1$ only. Finally, replacing the link to the leader with a link to another non-leader strictly increases the stretch to all but one peer in the network and therefore cannot be beneficial. □

Based on Lemma 23.12, we can consider the topology within each cluster in s to be fixed. It remains to show that no peer has an incentive to add, remove, or replace its inter-cluster links. The next lemma shows that peers in Π_y cannot unilaterally reduce their costs in s.

Lemma 23.13. *No peer in Π_y has an incentive to change its strategy, given that all other peers follow strategy s.*

Proof. By Lemma 23.12, no peer $v_y \in \Pi_y$ has an incentive to change its intra-cluster links. Furthermore, $\hat{v}_y$ does not benefit from switching its link from a leader peer to a non-leader peer, because this would only decrease the stretch to that particular peer, while increasing the stretch to all other peers (at least) in this cluster. It follows from Lemmas 23.10 and 23.9 that $\hat{v}_y$ must keep its links to $\hat{v}_j^a$ and $\hat{v}_z$, and hence, we must only consider that leader peers connect to leader peers.

We now show that no peer in Π_y can reduce this cost by deviating from its strategy in any other way.

Case 1: Some $\check{v}_y$ or $\hat{v}_y$ adds one of more additional links:
In the topology resulting from s, every peer in Π_y is connected with stretch at most $1 + 2\epsilon$ with all peers except from peers in Π_j^c (for all $C_j \in \mathcal{C}$) and peers in those literal-clusters to which $\hat{v}_z$ does not have a direct connection. With any additional link, a peer in Π_y can reduce its stretch to peers in exactly one of these clusters only. Hence, for every additional link, it holds that $\Delta_y(+) \geq -\frac{k(4.72+\epsilon)}{3.72} + \alpha + k > 0$. That is, adding additional links would increase the peer's cost.

Observe that because non-leader peers $\check{v}_y \in \Pi_y$ do not have inter-cluster links, Case 1 in combination with Lemma 23.12 implies that no $\check{v}_y$ can benefit

from changing its strategy.
Case 2: $\hat{v}_y$ changes its link from $\hat{v}_j^b$ to $\hat{v}_j^c$:
While the stretch to peers in Π_j^c is reduced, the stretch to peers in Π_j^b increases. The relative cost difference is $\Delta_y(\hat{v}_j^b \to \hat{v}_j^c) \geq -\frac{k(3+\epsilon)}{2.45} + \frac{(1.96+1.14)k}{2} > 0$.
Case 3: $\hat{v}_y$ removes its link from $\hat{v}_j^b$:
By removing such a link, $\hat{v}_y$ can save the link's cost α. On the other hand, the stretch to both Π_j^b and Π_j^c increase. Specifically, the shortest connection to peers in these clusters is now via $\hat{v}_j^a$ and $\hat{v}_j^b$, i.e., $\Delta_y(-\hat{v}_j^b) \geq -\alpha - k(1+\epsilon) - \frac{k(3+\epsilon)}{2.45} + \frac{(1.96+1.14)k}{2} + \frac{(1.96+1.14+1)k}{2.45} > 0$.

The only other thing that could potentially lead to an advantage for $\hat{v}_y$ is to replace a link $\hat{v}_j^b$ by some leader peer in Π_i^μ to which $\hat{v}_z$ is not connected, formally $\mu \neq A_I(x_i)$. Doing so clearly increases the stretch to peers in Π_j^b and Π_j^c, but like in Case 3, the shortest connection between $\hat{v}_y$ to peers in Π_j^c is via $\hat{v}_j^a$ and $\hat{v}_j^b$. In particular, this path has length at most $4.1 + \epsilon$, whereas the shortest path via a literal-cluster has length at least $1 - 2\delta + 1.72 + 1.48 = 4.2 - 2\delta$, which is larger. Hence, replacing one or more links to $\hat{v}_j^b$ by links to literal-peers reduces to Cases 1 and 3, respectively, and therefore cannot be worthwhile. Finally, it can be seen that any combination of the above cases cannot reduce the cost of any peer in Π_y either. □

Lemma 23.14. *No peer in Π_z has an incentive to change its strategy, given that all other peers follow strategy s.*

Proof. Again, we discuss the various cases and show that none of them is beneficial for a peer in Π_z. Recall that by Lemma 23.11, connecting to any clause-peer cannot improve the stretch to any other peer outside this clause. Furthermore, because A_I is a satisfying assignment, the topology of s contains a path of length at most $\epsilon + d(\Pi_z, \Pi_i^\mu) + d(\Pi_i^\mu, \Pi_j^c) + \epsilon = 3.2 + 2\epsilon$ between peers in Π_z and peers in Π_j^c, for *every* clause $C_j \in \mathcal{C}$. Consequently, connecting to a so far unconnected literal-peer cannot decrease the stretch to any clause-peer $v_j \in C_P$ in the system.

It follows from Lemma 23.12 that no peer $v_z \in \Pi_z$ has an incentive to change its intra-cluster links. Also, as shown in the proof of Lemma 23.13, no peer can benefit from connecting to a non-leader peer in the network, because this bears strictly higher costs than connecting to the corresponding leader peer of the same cluster. Hence, we only need to verify the cases in which peers in Π_z connect to leader peers.

In the following, we discuss the various cases how peers in Π_z could improve their situation and derive that none of them is actually beneficial.
Case 1: Some peer in Π_z adds an additional link to $\hat{v}_j^b$:
The reduction of the stretches to peers in Π_j^b and Π_j^c resulting from the additional link does not outweigh the link's cost. Specifically, we have $\Delta_z(+\hat{v}_j^b) \geq -\frac{k(3-2\delta+2\epsilon)}{2} + k - \frac{k(3.2+2\epsilon)}{2+\delta} + \frac{3k}{2+\delta} + \alpha \geq k(4\delta + 2\delta^2) > 0$. Notice that in the second term, the stretch to each of the k peers in Π_j^b is at least 1, and in the third term, the distance $3.2 + 2\epsilon$ holds because A_I is a satisfying assignment.

Case 2: Some peer in Π_z adds an additional link to $\hat{v}_j^c$:
Again, the stretches to Π_j^b and Π_j^c are not reduced enough to render the additional link worthwhile. In fact, the stretch to peers in Π_j^b is not reduced by the addition of this link, nor is the stretch to any other node in the network except from peers in Π_j^c (Lemma 23.11). It follows that $\Delta_z(+\hat{v}_j^c) \geq -\frac{k(3.2+2\epsilon)}{2+\delta} + k + \alpha = k(1.6\delta - 2\epsilon) > 0$.
Case 3: Some peer in Π_z adds an additional link to $\hat{v}_j^a$:
Clearly, this option is even worse than Cases 1 and 2.
Case 4: Some peer in Π_z adds an additional link to $\hat{v}_i^\mu$:
Adding a link to a literal-cluster that is not used in A_I reduces the stretch to peers in this cluster only, because there is already a short connection from Π_z to every Π_j^c through the literal-clusters $\Pi_i^{A_I(x_i)}$. Hence, $\Delta_z(+\hat{v}_i^\mu) \geq -\frac{k(2.72+2\epsilon)}{1.72} + k + \alpha > 0$.

Observe that because non-leader peers $\check{v}_z \in \Pi_z$ do not have inter-cluster links, Cases 1 to 4 in combination with Lemma 23.12 implies that no $\check{v}_z$ can benefit from changing its strategy.
Case 5: $\hat{v}_z$ replaces some $\hat{v}_i^{A_I(x_i)}$ by $\hat{v}_i^{1-A_I(x_i)}$:
Again, the new link to a previously unconnected literal-cluster cannot decrease the stretch to any clause-peer, because A_I is a satisfying assignment and $\hat{v}_z$ already had a path of length 3.2 to every $\hat{v}_j^c$ via some $\hat{v}_i^{A_I(x_i)}$. Furthermore, by a symmetry argument, the stretch cost gained by adding the link to $\hat{v}_i^{1-A_I(x_i)}$ is lost by removing the link to $\hat{v}_i^{A_I(x_i)}$. Therefore, $\Delta_y(\hat{v}_i^{A_I(x_i)} \rightarrow \hat{v}_i^{1-A_I(x_i)}) \geq 0$.
Case 6: $\hat{v}_z$ removes or replaces some $\hat{v}_i^{A_I(x_i)}$:
If $\hat{v}_z$ does not have a connection to any literal-cluster of a variable x_i, the resulting stretch to each peer in these two clusters is at least $\frac{3+\delta+1.48}{1.72}$. Because $\frac{k(4.48+\delta)}{1.72} > k(1+2\epsilon)+\alpha$, it follows that $\hat{v}_z$ must maintain at least one link to such a peer.

Any other possible strategy deviation can either be reduced to one of the above five cases or to Lemma 23.9, which concludes the proof. □

Having showed that peers in Π_y and Π_z have no incentive to deviate from s, it now remains to show that no other node can improve its situation either.

Lemma 23.15. *No top-layer peer can benefit from changing its strategy, given that all other peers follow s.*

Proof. First, by Lemma 23.12, it holds that no peer can improve its situation by adding, replacing, or removing a link within its cluster. Also, no node can benefit from connecting to a non-leader, as opposed to the leader peer in the same cluster. Both claims can be proven with exactly the same argument as in the proof of Lemma 23.13.

It is important to observe that in s, all top-layer peers are almost optimally connected with each other, either via the central cluster Π_c or because their respective clusters are neighbors in the graph. More specifically, the

stretch between any pair of top-layer peers in s is at most $1+2\epsilon$ (via the own cluster leader, $\hat{v}_c$, and the other cluster leader). Besides removing the final 2ϵ from these small stretches, adding additional links can only help in reducing the stretches to peers in Π_y and Π_z. By Lemma 23.9, no link between cluster leaders whose clusters have a distance of less than 1.48 can be removed from s. Hence, the possible strategy deviations by other nodes is actually limited.

Peers in Π_j^a: A peer $\hat{v}_j^a$'s link to $\hat{v}_y$ cannot be removed by Lemma 23.10. For every peer $v_j^a \in \Pi_j^a$, it further holds that building an additional link to $\hat{v}_z$ is too costly, $\Delta_j^a(+\hat{v}_z) \geq -\frac{k(2.96-2\delta+2\epsilon)}{2.45} + k + \alpha > 2kN\epsilon$. Hence, even if this additional link could reduce all other less than N stretches to top-level peers by the remaining 2ϵ, the cost of an additional link would still be too high.

Peers in Π_j^b: Peer $\hat{v}_j^b$ does not have a link longer than 1.48 in s and hence, cannot remove any of them. We show that neither building a link to $\hat{v}_y$ nor to $\hat{v}_z$ decreases the cost of any peer in Π_j^b. In the first case, we have $\Delta_j^b(+\hat{v}_y) \geq -\frac{k(1.96+1.14+2\epsilon)}{2} + k - \frac{k(3+\delta+2\epsilon)}{2} + \frac{k(3-2\delta)}{2} + \alpha > 2kN\epsilon$. As for the second case, $\Delta_j^b(+\hat{v}_z) \geq -\frac{k(1.96+1.14+2\epsilon)}{2} + \frac{k(3-2\delta)}{2} - \frac{k(3+\delta+2\epsilon)}{2} + k + \alpha > 2kN\epsilon$. Clearly, building both links is even less worthwhile.

Peers in Π_j^c: The potential strategy deviations that could decrease peer $\hat{v}_j^c$'s costs are to add a link to $\hat{v}_y$, to remove its link from $\hat{v}_z$, or to replace the link to $\hat{v}_z$ by a link to $\hat{v}_y$. However, none of these alterations are beneficial for $\hat{v}_j^c$ (or for any non-leader peer in Π_j^c in the case of link addition). First, it holds that $\Delta_j^c(+\hat{v}_y) \geq -\frac{k(3-\delta+2\epsilon)}{2.45} + k + \alpha > 2kN\epsilon$ and $\Delta_j^c(-\hat{v}_z) \geq -\alpha - k(1+2\epsilon) - \frac{k(3-\delta+2\epsilon)}{2.45} + \frac{3.2k}{2+\delta} + \frac{4.1k}{2.45} > 2kN\epsilon$. Also, switching the link from $\hat{v}_z$ to $\hat{v}_y$ is not helpful, $\Delta_j^c(\hat{v}_z \to \hat{v}_y) \geq \frac{3.2k}{2+\delta} - \frac{k(3-\delta+2\epsilon)}{2.45} > 2kN\epsilon$.

Peers in Π_i^μ: Each leader of a literal-cluster maintains a link to $\hat{v}_z$, and we show that they (as well as any non-leader peer in these clusters) do not have an incentive to change that strategy. It is clear that neither adding a link to $\hat{v}_y$ nor switching from $\hat{v}_z$ to $\hat{v}_y$ can be beneficial. In the first case, the stretch is reduced by at most 2ϵ by the additional link, which does not render the link cost α worthwhile. In the second case, the stretch is strictly increased. If $\hat{v}_i^\mu$ removes its link to $\hat{v}_z$ and connects via its neighboring literal-cluster, the stretches to both Π_y and Π_z increase. Particularly, we have $\Delta_i^\mu(-\hat{v}_z) \geq -\alpha - k(1+2\epsilon) + \frac{2.72k}{1.72} + \frac{k(3.72-2\delta)}{2.72-2\delta} > 2kN\epsilon$.

Peers in Π_c: Finally, peers in Π_c are connected with stretch at most 2ϵ to all peers in the network. To top-clusters, the connection is via links shorter than 1.48. As for the remaining two clusters, it is connected to $\hat{v}_z$ via one of the literal-clusters and to $\hat{v}_y$ via some $\hat{v}_j^a$. By the definition of ϵ and α, it is clear that no peer in Π_c can improve its strategy. □

By combining the three previous helper-lemmas, we can now state the main claim of this section.

Lemma 23.16. *If I is satisfiable, there exists a pure Nash equilibrium in $\mathcal{M}_I^k$.*

Proof. The lemma follows from the fact that by Lemmas 23.13, 23.14, and 23.15, no peer in the network has an incentive to change its strategy. Hence, s constitutes a pure Nash equilibrium. □

23.3.3 Non-satisfiable Instances

It remains to prove the other direction, that is, there exists no pure Nash equilibrium in the network if the underlying 3-SAT instance I has no satisfying assignment. We proceed by defining structural properties that any Nash equilibrium must fulfil, and show that the intersection of all these properties is empty. Besides the basic properties derived in Section 23.3.1, an important characteristic of any Nash equilibrium is the fact that exactly one peer in Π_z connects to *exactly one* literal-peer (either in Π_i^0 or Π_i^1) for every variable $x_i \in \mathcal{X}$.

Lemma 23.17. *In any Nash equilibrium, exactly one peer in Π_z connects to either a peer $v_i^1 \in \Pi_i^1$ or in $v_i^0 \in \Pi_i^0$, for every $x_i \in \mathcal{X}$.*

Proof. We have already shown in Lemma 23.14 (Case 6) that there must be a peer $v_z \in \Pi_z$ that has at least one link to a literal-peer of every variable. Furthermore, we know by Lemma 23.8 that no other peer in Π_z connects to the same cluster as v_z. Hence, we only need to show that in a Nash equilibrium no two peers in Π_z connect to both literal-clusters of the same variable.

Assume for the sake of contradiction that peers v_z and v_z' (potentially $v_z = v_z'$) maintain links to both Π_i^0 and Π_i^1 for some $x_i \in \mathcal{X}$. In this case, it would be worthwhile for one of the two peers to remove its link and replace it with a link to some peer in Π_j^c (if this link is not there, already), such that $x_i \in C_j$. By the definition of our special 3-SAT instance and the construction of G_I, we know that of the two literal-clusters, one, say Π_i^μ, has clause-cluster Π_j^c at distance 1.48, and the other literal-cluster, say $\Pi_i^{1-\mu}$, has two such close-by clause-clusters. Let v_z' be the peer that connects to cluster Π_i^μ (otherwise, replace v_z for v_z' for the remainder of the proof).

Assume for the first case that the length of the shortest path from v_z' to this Π_j^c without the link via Π_i^μ is 3.2 or longer. In this case, the change in v_z''s costs when switching from its link to literal-cluster Π_i^μ that has only a single close-by clause-cluster Π_j^c directly to a peer in Π_j^c is $\Delta_z(v_i^\mu \to v_j^c) \leq +\frac{k(2.72+2k\epsilon)}{1.72} - \frac{3.2k}{2+\delta} + \frac{k(2+\delta+2k\epsilon)}{2+\delta} < 0$. If the length of the path from v_z to Π_j^c is strictly shorter than 3.2, then the link to Π_i^μ can simply be dropped, resulting in a gain of $\Delta_z(-v_i^\mu) \leq -\alpha - k + \frac{k(1.72+2k\epsilon)}{1.72} + \frac{k(2.72+2k\epsilon)}{1.72} < 0$. Hence, v_z' is always better off not connecting to a literal-cluster if v_z already connects to a literal-cluster. From this, the claim follows. □

Lemma 23.17 is an important ingredient for the remainder of the proof, because it gives us a one-to-one correspondence between the connections of Π_z to literal-clusters, and an assignment of variables in the 3-SAT instance I. Also, note that when combining Lemma 23.17 with Lemma 23.11, it follows

that in a Nash equilibrium, peers in Π_y and Π_z cannot reduce their stretch to any peer in $V \setminus \{\Pi_j^a \cup \Pi_j^b \cup \Pi_j^c\}$ by connecting to one of the clause-peers of clause C_j.

Lemma 23.18. *If I is non-satisfiable, there exists no pure Nash equilibrium in $\mathcal{M}_I^k$.*

Proof. By Lemma 23.17, exactly one peer in Π_z connects to either the positive or negative literal-cluster of every variable x_i. Because there exists no satisfying assignment, it follows that regardless of how Π_z is connected to the literal-clusters, there must exist at least one clause C_{j^*} that is "not satisfied". In the resulting topology, this means that the path from a peer in Π_z to a clause-peer in $\Pi_{j^*}^c$ of this unsatisfied clause via any literal-cluster must be of length at least $d(\Pi_z, \Pi_i^\mu) + d(\Pi_i^\mu, \Pi_i^{1-\mu}) + d(\Pi_i^{1-\mu}, \Pi_{j^*}^c) = 4.2$. Particularly, every such path must include the additional distance of 1 between x_i^1 and x_i^0. In the sequel, we consider this *unsatisfied clause* C_{j^*} in more detail.

First, we show that in a Nash equilibrium, no peer $v_y \in \Pi_y$ establishes a link to $\Pi_{j^*}^c$. We distinguish two cases. In the first case, if some peer in Π_y already has a link to $\Pi_{j^*}^b$, then the cost reduction for v_y when omitting its link to $\Pi_{j^*}^c$ is $\Delta_y(-v_{j^*}^c) \leq -\alpha - k + \frac{k(3+2k\epsilon)}{2.45} < 0$. In the other case, the cost reduction when switching the link from $\Pi_{j^*}^c$ to a peer in $\Pi_{j^*}^b$ is at least $\Delta_y(v_{j^*}^c \to v_{j^*}^b) \leq -\frac{k(3-2\delta)}{2} + \frac{k(3+2k\epsilon)}{2.45} < 0$. That is, in either case it is beneficial for v_y not to connect directly to $\Pi_{j^*}^c$.

For the next step, we establish that in any Nash equilibrium, exactly one peer $v_z \in \Pi_z$ connects to either a peer in $\Pi_{j^*}^b$ or in $\Pi_{j^*}^c$. To see this, assume that no peer in Π_z establishes any links to peers in the two clusters. In this case (because there is no link from Π_y to $\Pi_{j^*}^c$, and because C_{j^*} is not satisfied), the sum of the stretches to peers in $\Pi_{j^*}^c$ is at least $\frac{k(4-2\delta)}{2+\delta} > k(1+2k\epsilon) + \alpha$. That is, $v_z \in \Pi_z$ can reduce its cost by connecting to $v_{j^*}^c$.

It remains to show that no peer in Π_z connects to $\Pi_{j^*}^a$, and particularly, that no two peers in Π_z simultaneously connect to both $\Pi_{j^*}^b$ or $\Pi_{j^*}^c$. Because there is at least one link from Π_z to either $\Pi_{j^*}^b$ or $\Pi_{j^*}^c$, it follows that a link to $\Pi_{j^*}^a$ can only reduce the stretch to peers in this particular cluster. However, the incurred cost exceeds the savings due to the reduced stretch, i.e., $\Delta_z(+v_{j^*}^a) = -\frac{k(2.96-2\delta+2k\epsilon)}{2.45} + \alpha + k > 0$. For the last case, assume that two peers v_z and v_z' (potentially the same) connect to both $\Pi_{j^*}^b$ and $\Pi_{j^*}^c$, respectively. Then, v_z' has an incentive to drop its link to $\Pi_{j^*}^c$: $\Delta_z(-v_{j^*}^c) = \frac{k(3+2k\epsilon)}{2+\delta} - k - \alpha < 0$. Hence, in any Nash equilibrium, there is exactly one link from Π_z to either $\Pi_{j^*}^b$ or $\Pi_{j^*}^c$, but not to both.

Studying the above rules, it can be observed that there remain only four possible sets of strategies for peers in Π_y and Π_z that could potentially result in a pure Nash equilibrium. The four cases can be distinguished by whether or not a peer in Π_y directly connects to $\Pi_{j^*}^b$, and by whether a peer in Π_z connects to $\Pi_{j^*}^b$ or $\Pi_{j^*}^c$. In the sequel, we analyze the four cases independently and show that none of them is actually a Nash equilibrium.

Case 1: Some peer $v_z \in \Pi_z$ connects to $v^b_{j^*}$:
In this case, some peer $v_y \in \Pi_y$ has an incentive to add a link to a peer in $\Pi^b_{j^*}$, because this significantly reduces its stretches to peers in $\Pi^b_{j^*}$ and $\Pi^c_{j^*}$. Specifically, v_y could reduce its cost by at least $\Delta_y(+v^b_{j^*}) \leq -\frac{k(3-2\delta)}{2} - \frac{k(4-2\delta)}{2.45} + \alpha + k(1+2k\epsilon) + \frac{k(3+2k\epsilon)}{2.45} < 0$.
Case 2: Peers $v_z \in \Pi_z$ and $v_y \in \Pi_y$ connect to $\Pi^b_{j^*}$:
In this case, the peer v_z can profit from switching its link to a peer in $\Pi^c_{j^*}$. Specifically, it holds that $\Delta_z(v^b_{j^*} \to v^c_{j^*}) \leq -\frac{3k}{2+\delta} + \frac{k(3-2\delta+2k\epsilon)}{2} < 0$.
Case 3: Some peer $v_z \in \Pi_z$ connects to $\Pi^c_{j^*}$:
Unlike in the previous case, v_z prefers switching its link from $\Pi^c_{j^*}$ to a peer in $\Pi^b_{j^*}$ in the absence of a link from Π_y to $\Pi^b_{j^*}$. By doing so, it can reduce its cost by $\Delta_z(v^c_{j^*} \to v^b_{j^*}) \leq \frac{k(3+2k\epsilon)}{2+\delta} - \frac{k(3+\delta)}{2} = k(-5\delta - \delta^2 + 4k\epsilon) < 0$.
Case 4: Some peer $v_z \in \Pi_z$ connects to $\Pi^c_{j^*}$ and some peer $v_y \in \Pi_y$ connects to $\Pi^b_{j^*}$:
In this configuration, peer v_y benefits from removing its link to $\Pi^b_{j^*}$. The decrease of its costs is $\Delta_y(-v^b_{j^*}) < -\alpha - k + \frac{k(3.1+2k\epsilon)}{2} < 0$.
Finally, since none of these four cases is a Nash equilibrium, the proof is concluded. □

Chapter 24

When Selfish Meets Evil: Byzantine Players among Selfish Agents

The previous chapter has explored the impact of *selfishness* on a system's efficiency and stability. But what happens if not every node is selfish? A system consisting of selfish utility-optimizing agents may also have to cope with malicious *Byzantine adversaries* who seek—independently of their own cost—to degrade the utility of the entire system, to attack correctness of certain computations, or to cause endless changes which render the system instable.

In this chapter, we aim at combining selfishness and Byzantine behavior. In particular, we consider a system of selfish individuals whose only goal is to optimize their own benefit, and add malicious players who attack the system in order to deteriorate its overall performance. Or asked succinctly: What is the impact of the Byzantine players on a selfish system's efficiency?

The question of Byzantine threats in system consisting of selfish agents appears to be of actuality in many research fields. Examples in computer science include *Internet viruses* or *Denial of Service attacks* where some players aim at destructing systems which otherwise typically consist of utility-maximizing players. However, such phenomena might also arise in economic or sociological environments. For instance, one can imagine a set of companies competing on a market, selfishly seeking to maximize their personal gains. However, there might be one or two companies run by "terrorists" whose goal is to destabilize the economic system.

In order to capture these questions formally, Section 24.3 introduces the *Price of Malice* of selfish systems. The Price of Malice is a ratio that expresses how much the presence of malicious players deteriorates the social welfare of a system consisting of selfish players. More technically, the Price of Malice is the ratio between the social welfare or performance achieved by a selfish

system containing a number of Byzantine players, and the social welfare achieved by an entirely selfish society.

It is interesting to compare the Price of Malice with the notion of the Price of Anarchy. The Price of Anarchy captures the degradation of a socially optimal performance of a system due to selfish behavior of its users or participants. That is, the Price of Anarchy relates the social welfare generated by players acting in an egoistic manner to an optimal solution obtained by perfectly collaborating participants. The Price of Malice's reference point, on the other hand, is not a socially optimal welfare, but the welfare achieved by an entirely selfish system.

The Price of Anarchy and the Price of Malice can therefore be considered as two orthogonal measures that describe properties of distributed, socio-economic systems. Specifically, a system may have a small Price of Anarchy, but a large Price of Malice, and vice versa. The fact that a system has a large *Price of Anarchy* indicates that it is necessary to design mechanisms (such as taxes, payment schemes, or coordination mechanisms) that forces players to collaborate more efficiently. However, it is much more difficult to improve (or *repair*) systems having a large *Price of Malice*, since Byzantine players may not respond to any rules or (financial) incentives. Often, the only solution is to defend the system against malicious intruders, or at least to ensure that the number of malicious players in the system remains small.

The Price of Malice crucially depends on the amount of *information* the selfish players have about the presence and behavior of the Byzantine players, and how they respond to this information. In other words, the utility function which finally defines the selfish players' reaction depends on how they *subjectively* perceive and judge the threat of Byzantine players. That is, the utility of selfish players is computed using the *perceived expected cost* rather than the unknown actual cost. Interestingly, it can be shown that if players are risk-averse, the presence of Byzantine players may actually *improve* the social welfare compared to a situation where there are no Byzantine players at all. In other words, there are situations in which the selfish players' increased willingness to collaborate in the view of higher risks outweighs the cost caused by malicious, Byzantine players.

In this chapter, we investigate a concrete example where selfish and Byzantine players interact. In this simple game, which was inspired by [15], there is a network of nodes, where each node can choose between paying for inoculation, or risking to get infected by a virus. After the nodes have made their choices, a virus starts at some random node and propagates iteratively to all neighboring nodes which are not inoculated. For this *virus inoculation game*, Section 24.4 presents bounds on the Price of Malice for different types of selfish nodes.

From a strictly game theoretical point of view, Byzantine players can be regarded as a special species of selfish players whose valuation is negatively correlated to the social welfare of the entire system. Strictly speaking, studying Byzantine players in selfish games can therefore be accomplished within the classic game theoretic framework. However, given a specific game setting, the *semantic* meaning or purpose of such Byzantine players is different from regular selfish players.

24.1 Related Work

Security and robustness of distributed systems against Byzantine faults have been of prime importance and an active field of research for many years. Possibly the most well-known problem in this context has been that of reaching *consensus* among distributed parties. Possibility and impossibility results on the Byzantine consensus problem have been achieved in a variety of models and settings. Classic work in the synchronous and asynchronous case includes [70, 92, 157, 213]. In addition to the consensus problem, the distributed computing community has come up with results and solutions for a wide variety of other problems with Byzantine faults. Examples are clock synchronization [233], broadcast [142, 219], or quorum systems [172]. All of the above works assume that non-Byzantine players (or processes) are benevolent and attempt to reach a common goal. Finally, Byzantine behavior is subject to intensive research in cryptography.

By combining this thread of research with *selfishness*, the content of this chapter is related to the notions of *fault tolerant implementation* introduced by Eliaz [74] and of *BAR fault tolerance* introduced by Aiyer et al. [7]. In [74], implementation problems are investigated where there are k faulty players in the population, but neither their number nor their identity is known. A planner's objective then is to design an equilibrium where the non-faulty players act according to his rules. In [7], the authors describe an asynchronous state machine replication protocol which tolerates Byzantine, Altruistic, and Rational behavior. Interestingly, they find that the presence of Byzantine players can simplify the design of protocols if players are risk-averse.

There exists other work on game theoretic systems in which not every participating agent acts in a rational and selfish way. In Stackelberg theory [226], for instance, the model consists of a set of selfish players, but a certain fraction of the entire population (or in routing games: a certain fraction of the entire flow) is controlled by a global *leader*. The rest of the users acts selfishly and the resulting equilibrium consisting of leader-controlled and selfish players is called the *Stackelberg equilibrium*. The goal of the global leader is to devise a strategy that induces an optimal or near optimal Stackelberg equilibrium. Stackelberg equilibria in network related settings have for instance been studied in [143, 153, 207].

24.2 Virus Inoculation Game

In [15], Aspnes, Chang and Yampolskiy model *virus inoculation* as a game with n strategic players each of which corresponds to a node in graph G. In our case, we focus on the topology of an undirected grid $G[r, c]$ consisting of r rows and c columns.[1] Henceforth, we refer to the upper left corner of the grid as $G[0, 0]$, i.e., indices start with 0.

Each node v_i has two choices: either do nothing and risk infection by a virus, or inoculate itself by installing anti-virus software. For a node, in-

[1] Our results can be generalized to other highly regular, low-dimensional graphs such as the two-dimensional torus, i.e., a grid that wraps around at the boundaries.

stalling the anti-virus software has the obvious advantage that it becomes immune against infection. On the other hand, the process of installing the software entails a cost in terms of money and/or time. Hence, a strategic player may or may not opt for inoculation depending on which choice maximizes its own utility.

The nodes' choices can be summarized by a strategy profile $\overrightarrow{a} \in \{0,1\}^n$, where $a_i = 1$ signifies that node v_i installs the anti-virus software, and $a_i = 0$ that it does not install it. We call nodes v_i with $a_i = 1$ *secure*, and denote the set of secure nodes as $I_{\overrightarrow{a}}$. After the nodes have made their choices, the adversary picks some node uniformly at random as a starting point for infection. Infection then propagates through the network graph and infects all non-secure nodes that are in the same non-secure connected component as the starting point for infection. Technically, we associate an *attack graph* $G_{\overrightarrow{a}} = G \setminus I_{\overrightarrow{a}}$ with $\overrightarrow{a}$. It is essentially the network graph in which all secure nodes and their incident edges are removed.

The associated costs are as follows: installing anti-virus software on a selfish node entails an inoculation cost of 1 at this node. If a selfish node does not inoculate and becomes infected, it suffers a loss equal to L. Therefore, the cost of a selfish node v_i can be summarized as follows:

$$cost_i(\overrightarrow{a}) = a_i + (1 - a_i) \cdot L \cdot \frac{k_i}{n}, \tag{24.1}$$

where k_i/n is the probability that node v_i is infected, conditioned on the event that it does not install the anti-virus software. Thereby, k_i is the size of the connected component containing v_i in $G_{\overrightarrow{a}}$. Finally, the *social cost* of a strategy profile $\overrightarrow{a}$ is the sum of all individual costs, i.e., $Cost(\overrightarrow{a}) = \sum_{v_j \in S} cost_j(\overrightarrow{a})$, where S denotes the set of all selfish players. When the strategy profile $\overrightarrow{a}$ is clear from the context, we sometimes use abbreviations $cost_i$ and $Cost$ to denote individual cost and social cost, respectively.

In [15], several results on the virus inoculation game have been proven. Specifically, [15] provides a characterization of Nash equilibria and establishes that the problem of finding either the most or least expensive equilibrium is NP-hard. Finally, [15] also studies the centralized optimization problem induced by the virus inoculation game and presents an $O(\log^2 n)$ approximation algorithm for this problem.

24.3 Byzantine Game Theoretic Model

In order to analyze the impact of malicious players on the selfish system, we enrich the virus inoculation game of the previous section with malicious Byzantine players. Formally, there are n nodes in the network. Of these n nodes, b are malicious *Byzantine nodes* that do not strive for minimizing their own costs. Instead, the goal of these Byzantine nodes is to deteriorate the overall system performance as much as possible, i.e., to maximize the resulting social cost of the solution. The remaining $s := n - b$ nodes are *selfish* and aim at maximizing their own utility. The sets of Byzantine and

selfish players are denoted by B and S, respectively. It holds that $b := |B|$, $s := |S|$, and $n = s + b$.

While selfish nodes behave as discussed in Section 24.2, we assume that the Byzantine nodes pursue the following strategy: they claim to be inoculated (i.e., they proclaim their strategy to be $a_i = 1$), but actually they are not. In order to emphasize that Byzantine nodes are only seemingly secure, we denote the set of really inoculated and secure selfish nodes by $I_{\overrightarrow{a}}^{self}$. The attack graph resulting from strategy profile $\overrightarrow{a}$ is then $G_{\overrightarrow{a}} = G - I_{\overrightarrow{a}}^{self}$. This is the network graph without secure, selfish nodes, but including all Byzantine nodes. We can therefore define the individual cost incurred at a selfish node $v_i \in S$ as follows.

Definition 24.1 (Actual Individual Cost). *The* (actual) individual cost $cost_i(\overrightarrow{a})$ *of a node* $v_i \in S$ *is defined as*

$$cost_i(\overrightarrow{a}) := a_i + (1 - a_i) \cdot L \cdot \frac{k_i}{n},$$

where k_i *is the size of the connected component of node* v_i *in the attack graph* $G_{\overrightarrow{a}}$.

Notice that in spite of its being equivalent to the corresponding definition in Section 24.2, we call this cost *actual individual cost*. This is to emphasize the fact that selfish players may *not know* about the existence of Byzantine players, and therefore, they are unable to compute their actual individual cost. Even if they are aware of the malicious players' existence, they might not know the Byzantine players' exact locations or strategies. In other words, with the addition of Byzantine players, selfish nodes no longer have a *perfect knowledge* about the network and its nodes' choices.

In case of imperfect information, a node might deal with its uncertainty in different ways. For example, a node might be risk averse and act in a conservative manner. These observations imply that before the location and strategies of Byzantine players are revealed (i.e., before the virus infection occurs), a selfish player v_i experiences a *perceived individual cost* $\widehat{cost}_i(\overrightarrow{a})$. This perceived cost can differ from the *actual individual cost* $cost_i(\overrightarrow{a})$ a node eventually has to pay.

Definition 24.2 (Perceived Individual Cost). *Assume a selfish game with Byzantine players in which selfish players have imperfect knowledge about the existence, location, or the strategy of Byzantine players. In this case, the* perceived individual cost $\widehat{cost}_i(\overrightarrow{a})$ *of a selfish player* v_i *captures the cost expected by player* v_i *given its knowledge about the Byzantine players. This cost depends on the underlying model.*

The strategic decisions of selfish players can only be based on the *perceived cost* (not on their actual individual costs), because the actual individual cost can only be computed once the locations and strategies of Byzantine players are revealed. In the sequel, we study the following two basic models.

Definition 24.3 (Oblivious Model). *In the* oblivious model*, selfish players are not aware of the existence of Byzantine players. That is, selfish players assume that all other players in the system are selfish as well.*

Definition 24.4 (Non-Oblivious Model). *In the* non-oblivious model*, selfish players know about the existence of Byzantine players. Specifically, we assume that every selfish player knows b, the number of Byzantine players in the system, but it does not know about these players' exact locations or strategies. Moreover, we assume that selfish players are highly risk-averse in the sense that they aim at minimizing their maximal individual cost.*[2] *Let $\mathcal{D}$ be the set of possible distributions of Byzantine players among all players. A selfish player v_i experiences a perceived individual cost of*

$$\widehat{cost}_i(\overrightarrow{a}) := \max_{d \in \mathcal{D}} \{cost_i(\overrightarrow{a}, d)\},$$

where $cost_i(\overrightarrow{a}, d)$ denotes the actual costs of v_i if the Byzantine players are distributed according to $d \in \mathcal{D}$.

In the virus inoculation game, and in an oblivious model, the perceived cost is typically smaller than the actual cost: A node $v_i \in S$ does not take into consideration Byzantine nodes which may increase the size of v_i's attack components. In the non-oblivious risk-averse model, on the other hand, a node actually overestimates its expected actual cost by considering a worst-case scenario: A selfish player assumes that the Byzantine nodes are—from its individual point of view—distributed in a worst-case fashion among all players. Therefore, the perceived individual cost may be larger than the actual cost.

Since our goal is to understand the impact of malicious behavior on a system of selfish players, the cost of Byzantine players is not included in the social cost. If it was, it would in general be easy for Byzantine players to arbitrarily deteriorate the social welfare of a system by simply increasing its own cost as much as possible. Moreover, as Byzantine players are malicious anyway, there is no particular reason why the overall system should care about these players' costs.

Formally, the total *social cost* $Cost(\overrightarrow{a})$ of a strategy is defined as the sum of the (actual) individual costs of all selfish players. Since each node in the same connected component of $G_{\overrightarrow{a}}$ has the same probability of infection, the l_i selfish nodes in the i-th attack component face a loss of $l_i \cdot (Lk_i/n)$ if the component is infected.

Definition 24.5 (Social Cost). *The* social cost *is given by the sum of the actual individual costs of selfish players*

$$Cost(\overrightarrow{a}) \quad = \quad \sum_{j \in \mathcal{S}} cost_j(\overrightarrow{a}) \quad = \quad \underbrace{|I_{\overrightarrow{a}}^{self}|}_{inoculation\ cost} \quad + \quad \underbrace{\frac{L}{n} \sum_{i=1}^{l} k_i l_i}_{infection\ cost} \quad ,$$

[2] Numerous other models would also be interesting. For instance, one could study a non-oblivious risk-friendly model in which every selfish player computes its perceived cost by assuming a random distribution of the b Byzantine players in the system.

where $k_1, k_2, \ldots, k_l$ are the sizes of the components in $G_{\overrightarrow{a}}$, and $l_1, l_2, \ldots, l_l$ are the sizes of the same components without counting the Byzantine nodes. We refer to the cost due to inoculation as the inoculation cost $Cost_{inoc}$*, and to the cost due to the virus infections as the* infection cost $Cost_{inf}$*.*

As customary, the social cost of a setting where all nodes perfectly collaborate, i.e., where there are neither selfish nor Byzantine nodes, is called the *social optimum*.

Definition 24.6 (Optimal Social Cost). *The* optimal social cost $Cost_{OPT}$ *is the sum of all the players' actual individual costs in case of perfect collaboration.*

The classic Nash equilibrium describes a situation where no selfish node has an incentive to unilaterally change its strategy. This definition can be extended to incorporate Byzantine nodes as follows. The *Byzantine Nash equilibrium* (BNE) describes a configuration where no selfish player can reduce its *perceived* cost by changing its strategy, given the strategies of all other players are fixed.[3]

Definition 24.7 (Byzantine Nash Equilibrium (BNE)). *Let $\overrightarrow{a}[i|x]$ be the strategy vector that is identical to $\overrightarrow{a}$ except for the i-th component a_i which is replaced by x. In a* Byzantine Nash equilibrium*, no selfish player $v_i \in \mathcal{S}$ has an incentive to change its strategy if the strategies of all other selfish players are fixed, i.e.,*

$$\forall v_i \in \mathcal{S} : \widehat{cost}_i(\overrightarrow{a}) \leq \widehat{cost}_i(\overrightarrow{a}[i|a'_i]),$$

for every possible strategy a'_i.

While the Byzantine Nash equilibrium must be defined by the *perceived* individual costs, the resulting social cost is determined by the *actual* costs. After all, it is the actual individual costs that players will eventually have to pay. The social cost of the *worst Byzantine Nash Equilibrium* of a problem instance I with b Byzantine players is denoted by $Cost_{BNE}(I, b)$.

It is well-known that selfish and Byzantine players often interact in a manner that yields suboptimal solutions. The degree of degradation resulting from selfish and Byzantine players compared to the social optimum is captured by the *Price of Byzantine Anarchy*.

Definition 24.8 (Price of Byzantine Anarchy $PoB(b)$). *The* Price of Byzantine Anarchy *captures how much worse a Byzantine Nash equilibrium can be compared to a collaborative optimal solution. More formally, in a scenario with b Byzantine players, the Price of Byzantine Anarchy $PoB(b)$ is the ratio between the worst-case social cost of a Byzantine Nash equilibrium divided by the minimal social cost, i.e., for all problem instances I,*

$$PoB(b) = \max_I \frac{Cost_{BNE}(I, b)}{Cost_{OPT}(I)}.$$

[3]Notice that the Byzantine Nash equilibrium cannot be defined with actual individual costs, because these are not known to the players.

Note that in the absence of any Byzantine players—i.e., if the system consists of purely selfish players only—, the Price of Byzantine Anarchy is equivalent to the Price of Anarchy (PoA). Specifically, we have $PoA = PoB(0)$.

With these definitions, we are finally ready to define the *Price of Malice*, which describes the degree of sub-optimality resulting from malicious Byzantine players in an otherwise selfish system. A high Price of Malice indicates that an economic system is particularly vulnerable to malicious or terrorist attacks. On the other hand, if the Price of Malice is low, the system consisting of selfish players is stable enough to tolerate malicious participants. Clearly, the degree of degradation may depend on the number of Byzantine players in the game. Hence, the Price of Malice is a function of b.

Definition 24.9 (Price of Malice $PoM(b)$). *The Price of Malice captures the ratio between the worst Byzantine Nash Equilibrium with b malicious players and the Price of Anarchy in a purely selfish system. Formally,*

$$PoM(b) = \frac{PoB(b)}{PoB(0)}.$$

As we discuss in Section 24.4.4, the inverse of the Price of Malice may also be considered as the game's *Fear Factor* $\Psi(b)$. That is, $\Psi(b) := 1/PoM(b)$.

24.4 Price of Malice

In order to derive results on the Price of Malice, we first establish structural properties of Nash equilibria and the social optimum in the virus inoculation game. We begin with a simple characterization of Nash equilibria when there are no Byzantine nodes. The proof of the following lemma follows from the analogous lemma in [15].

Lemma 24.1. *In a pure Nash equilibrium $\overrightarrow{a}$, it holds that (a) Every component in the attack graph $G_{\overrightarrow{a}}$ has size at most n/L. (b) Inserting any secure node into $G_{\overrightarrow{a}}$ yields a component size of at least n/L.*

Lemma 24.1 implies that if $L \geq n$, all nodes will inoculate in the Nash equilibrium. In the following, it is therefore assumed that $L < n$.

24.4.1 Social Optimum

If the inoculation strategies of the individual nodes are planned by a benevolent centralized coordinator, the welfare of the system can be maximized. Throughout this section, perceived costs equal actual costs because when studying the social optimum, there are no Byzantine players, i.e., $b = 0$ and therefore $s = n$.

Theorem 24.2. *The optimal social cost if all players in S act altruistically is $Cost_{OPT} \in \Theta(s^{2/3}L^{1/3})$. More specifically,*

$$\frac{1}{3}\sqrt{\pi} \cdot s^{2/3}L^{1/3} \;\leq\; Cost_{OPT} \;\leq\; 4s^{2/3}L^{1/3}.$$

Proof. We prove the upper and lower bound in turn.

Lower Bound: If all nodes collaborate to achieve the optimal solution, it holds that $l_i = k_i$ and hence, the social cost is given by

$$Cost \;=\; |I_{\overrightarrow{a}}| + \frac{L}{n}\sum_{i=1}^{l} k_i^2,$$

where $|I_{\overrightarrow{a}}|$ is the number of inoculated nodes, and the k_i's are the sizes of the components in the attack graph. This sum is minimized when all k_i are of equal size, say size K. While each secure node has a cost of 1, every other node has an expected cost of $L \cdot K/n$. Hence, setting $\gamma := |I_{\overrightarrow{a}}|$ and because $s = n$, the optimal social cost is

$$Cost_{OPT} \;\geq\; \gamma + (s-\gamma)\left(\frac{LK}{s}\right). \tag{24.2}$$

A relationship between γ and K follows from a simple geometric argument: If a component in the attack graph is of size K, the number of inoculated nodes at the components border must be at least $2\pi\sqrt{\frac{K}{\pi}} = 2\sqrt{\pi K}$, because this is the circumference of a disk with volume K. As the total number of such components is at least $\frac{s-\gamma}{K}$ and each inoculated node can be on the border of at most 2 components, γ can be expressed as

$$\gamma \;\geq\; \frac{s-\gamma}{K} \cdot 2\sqrt{\pi K} \cdot \frac{1}{2} \;=\; (s-\gamma)\sqrt{\frac{\pi}{K}}.$$

When solving this inequality for γ, it follows that $\gamma \;\geq\; s \cdot \frac{\sqrt{\pi/K}}{1+\sqrt{\pi/K}}$. On the other hand, it can be observed that in the optimal solution, for $s > L$, no node is inoculated if all its four neighbors are inoculated. From this, it can be derived that in an optimal solution, $\gamma \leq \frac{s}{2}$. Plugging these two bounds into Inequality (24.2), the optimal social cost is at least

$$Cost_{OPT} \;\geq\; s \cdot \frac{\sqrt{\pi/K}}{1+\sqrt{\pi/K}} + \frac{LK}{2}.$$

The first term of the above expression is monotonously decreasing in K in the range $0, \ldots, s$, whereas the second one is monotonously increasing. Therefore, taking the minimum of the two terms for a specific K yields a lower bound on $Cost_{OPT}$. When setting

$$K := \frac{2}{3}\sqrt{\pi} \cdot \left(\frac{s}{L}\right)^{2/3},$$

the second term yields $\frac{1}{3}\sqrt{\pi} \cdot s^{2/3}L^{1/3}$. The first term evaluates to $\frac{\sqrt{3/2}\cdot\sqrt[4]{\pi}}{1+\sqrt{3/2}\cdot\sqrt[4]{\pi}} \cdot s^{2/3}L^{1/3} \;>\; \frac{1}{3}\sqrt{\pi} \cdot s^{2/3}L^{1/3}$. Consequently, we obtain the following lower

bound on the cost of the social optimum:

$$Cost_{OPT} \quad \geq \quad \frac{1}{3}\sqrt{\pi} \cdot s^{2/3} L^{1/3} \ \in \ \Omega(s^{2/3} L^{1/3}).$$

Upper Bound: We construct a solution that is asymptotically optimal and proves tightness of the above lower bound. Given an arbitrary grid $G[r, c]$, we inoculate the nodes as follows. Let $K := (s/L)^{2/3}$. We secure all nodes in the columns $G[\cdot, i\sqrt{K}]$ for $i \in \{1, ..., \lfloor c/(\sqrt{K}+1)\rfloor\}$ and rows $G[i\sqrt{K}, \cdot]$ for $i \in \{1, ..., \lfloor r/(\sqrt{K}+1)\rfloor\}$. Consequently, all attack components are of size at most $\sqrt{K} \times \sqrt{K} = K$ as illustrated in Figure 24.1 (left). Hence, the total infection cost is at most $L \cdot (s - |I_{\vec{a}}|)\frac{K}{s} < LK = s^{2/3}L^{1/3}$.

It remains to bound the inoculation cost. In an ideal setting where the components perfectly fit into $G[r, c]$ without leftovers, it holds that for each component of size K in the attack graph, there are exactly $2\sqrt{K}+1$ inoculated nodes. Let X denote the number of components. It holds that $X \cdot (K + 2\sqrt{K} + 1) = s$ and therefore, when plugging in the definition of K, $X = s/[(\frac{s}{L})^{2/3} + 2(\frac{s}{L})^{1/3} + 1]$. The number of inoculated nodes γ is at most

$$\begin{aligned}
\gamma \quad &\leq \quad X \cdot (2\sqrt{K}+1) \leq \frac{s(2\sqrt{K}+1)}{\left(\frac{s}{L}\right)^{2/3} + 2\left(\frac{s}{L}\right)^{1/3} + 1} \\
&< \quad s^{1/3}L^{2/3} \cdot \left(2\left(\frac{s}{L}\right)^{1/3} + 1\right) = 2s^{2/3}L^{1/3} + s^{1/3}L^{2/3} \\
&\leq \quad 3s^{2/3}L^{1/3}.
\end{aligned}$$

Combining the infection and inoculation costs, we can bound the optimal social cost by

$$Cost_{OPT} < s^{2/3}L^{1/3} + 3s^{2/3}L^{1/3} = 4s^{2/3}L^{1/3}.$$

□

24.4.2 Price of Anarchy

The Price of Anarchy compares the social cost of the worst Nash equilibrium (without Byzantine nodes) to the minimal social cost. In the upcoming section, we will first compute $Cost_{NE}$, which is the maximal cost of any Nash equilibrium. Together with the bound for the social optimum in Section 24.4.1, the Price of Anarchy will follow.

Lemma 24.3. *The social cost of the worst Nash equilibrium is* $Cost_{NE} = \Theta(s)$.

Proof. First, we show that $Cost_{NE} = \Omega(s)$. Consider a grid $G[s/L, L]$ consisting of an even number of L rows of size s/L. Assume that columns $G[\cdot, 2i]$ for $i \in \{0, 1, ..., L/2 - 1\}$ consist of insecure nodes only, while all

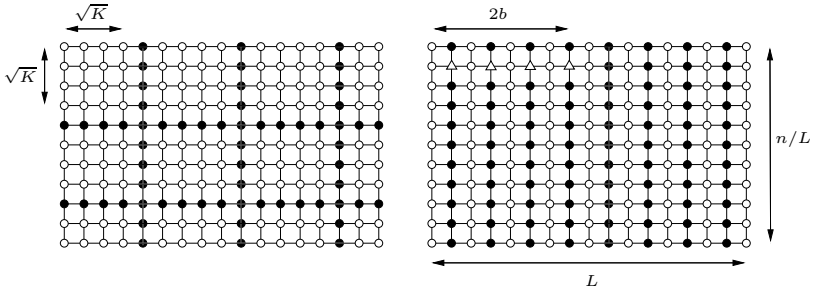

Figure 24.1: *Left:* Upper bound for social optimum. White nodes are insecure, black nodes are secure. *Right:* Byzantine Nash equilibrium for $G[n/L, L]$ for the oblivious model. Insecure Byzantine nodes are denoted by white triangles. They are located in a way that may yield an attack component of size $(b+1)n/L + b$.

nodes in the remaining rows are secure. Since all attack components have size s/L, according to Lemma 24.1, this situation constitutes a Nash equilibrium. Observe that every second row is inoculated, engendering an inoculation cost of $s/2$. Moreover, with probability $1/2$, the virus starts at an insecure node, yielding infection cost $s/L \cdot L$. The social cost is therefore $Cost_{NE} = s/2 + 1/2 \cdot s/L \cdot L = s$.

It remains to show that $O(s)$ is an upper bound for any Nash equilibrium. Since at most each of the $s = n$ nodes can be inoculated, the inoculation cost cannot exceed s. By Lemma 24.1, we also know that the infected component's size is at most s/L, entailing a total infection cost of at most s as well. Hence, $Cost_{NE} \leq 2s$, and the claim holds. □

By Theorem 24.2 and Lemma 24.3, we get the following result.

Theorem 24.4. *For the Price of Anarchy (PoA), it holds that*

$$\frac{1}{4} \cdot \left(\frac{s}{L}\right)^{1/3} \leq PoA \leq \frac{6}{\sqrt{\pi}} \cdot \left(\frac{s}{L}\right)^{1/3}$$

Proof. As for the upper bound, it holds that

$$PoA = \frac{Cost_{NE}}{Cost_{OPT}} \leq \frac{2s}{\frac{1}{3}\sqrt{\pi} \cdot s^{2/3} L^{1/3}} \leq \frac{6s^{1/3}}{\sqrt{\pi} \cdot L^{1/3}}$$

and as for the lower bound, we have $PoA \geq \frac{s}{4 \cdot s^{2/3} L^{1/3}}$. □

24.4.3 Oblivious Model

We begin our study of the Price of Malice in the *oblivious model* in which players are clueless about the existence of Byzantine players in the system.

As nodes consequently underestimate the attack components' sizes, it follows that the nodes' perceived individual costs are smaller than the actual individual costs. Not surprisingly, the social costs increase with the number of Byzantine nodes.

Lemma 24.5. *In the oblivious model, the social cost is at least* $Cost_{BNE} \in \Omega(s + \frac{nb^2}{L})$ *for* $b < \frac{L}{2} - 1$*, and* $Cost_{BNE} \in \Omega(sL)$ *otherwise.*

Proof. Consider again a grid $G[n/L, L]$ with n/L rows and L columns, where every second column consists of secure nodes only. For simplicity, let L be even. Suppose that in the first b secure columns there is one Byzantine node each, see Figure 24.1 (middle). In case $b \geq \frac{L}{2} - 1$, every secure column that separates two insecure columns contains one Byzantine node. The remaining Byzantine nodes can be placed at arbitrary places in the secure columns. Because selfish nodes are not aware of the existence of Byzantine nodes in the network, the perceived cost is $\widehat{cost}_i = 1$ for inoculated nodes, and $\widehat{cost}_i = \frac{n/L}{n} \cdot L = 1$ for the other selfish nodes. Hence, the situation constitutes a *Byzantine Nash equilibrium*.

For computing the social costs of this Byzantine Nash equilibrium, we distinguish two cases, depending on whether the number of Byzantine nodes is smaller than $\frac{L}{2}-1$ or not. For the first case, assume that $b \geq \frac{L}{2}-1$. Because there is at least one Byzantine node in every secure column that separates two insecure columns has least one Byzantine node, all selfish and Byzantine players form one large attack component. Consequently, each insecure selfish node $i \in S$ is infected with probability 1 and therefore $Cost_{BNE} \geq s \cdot L$.

For the second case, assume that $b < \frac{L}{2} - 1$. Each of the first secure columns contains exactly one Byzantine node. Since L is even, there are $s/2-b$ secure nodes, and hence the inoculation cost is $s/2 - b$. With probability $((b+1)n/L+b)/n$, the infection starts at an insecure or a Byzantine node of an attack component of size $(b+1) \cdot n/L$, yielding a cost of $(b+1) \cdot n/L \cdot L = n(b+1)$. Moreover, with probability $(s/2-(b+1)n/L)/n$, an insecure column of size n/L is hit. Thus, for $b < \frac{L}{2} - 1$, we get the following lower bound on the social cost:

$$\begin{aligned} Cost_{BNE} &= \left(\frac{s}{2} - b\right) + \frac{\frac{(b+1)n}{L} + b}{n} \cdot n(b+1) + \frac{\frac{s}{2} - (b+1)\frac{n}{L}}{n} \cdot \frac{n}{L} \cdot L \\ &= s + \frac{nb^2}{L} + \frac{nb}{L} + b^2 \in \Omega\left(s + \frac{nb^2}{L}\right). \end{aligned}$$

□

Lemma 24.6. *In the oblivious model, the social cost is at most* $Cost_{BNE} \in O\left(\min\{sL, s + \frac{b^2 n}{L}\}\right)$.

Proof. Since at most every selfish node can be inoculated, it is clear that $Cost_{inoc} = O(s)$. It remains to study the infection cost. The infection cost of a node in some component i is L times the probability of this component

being hit by the virus, i.e., $L \cdot k_i/n$. Hence, the total infection cost is given by

$$Cost_{inf} \;=\; \sum_i l_i \cdot \frac{k_i}{n} \cdot L \;=\; \frac{L}{n} \sum_i l_i \cdot k_i,$$

where k_i is the size of the attack components (including Byzantine nodes), and l_i is the number of selfish nodes in this component. In order to upper bound $Cost_{inf}$, let S_{Byz} denote the set of components in the attack graph which contain at least one Byzantine node, and let $S_{\overline{Byz}}$ be the remaining components. We can rewrite the equation above as

$$Cost_{inf} \;=\; \frac{L}{n} \cdot \left[\sum_{i \in S_{Byz}} l_i \cdot k_i + \sum_{i \in S_{\overline{Byz}}} l_i \cdot k_i \right],$$

that is, we consider the infection cost of components with at least one Byzantine node separately from the remaining "Byzantine-free" components. In the following, let

$$Cost_{inf}^{Byz} := \frac{L}{n} \sum_{i \in S_{Byz}} l_i k_i \qquad Cost_{inf}^{\overline{Byz}} := \frac{L}{n} \sum_{i \in S_{\overline{Byz}}} l_i k_i.$$

We have to prove that neither $Cost_{inf}^{Byz}$ nor $Cost_{inf}^{\overline{Byz}}$ exceeds $O(s + \frac{b^2 n}{L})$.

As we have shown in the proof of Lemma 24.3 in Section 24.4.2, the total infection cost of a network consisting only of selfish nodes cannot exceed s. Because in our case nodes are oblivious about the existence of Byzantine nodes, attack components without Byzantine nodes behave like in an entirely selfish environment. Therefore, $Cost_{inf}^{\overline{Byz}} \in O(s)$.

It remains to compute the infection cost of those attack components which include at least one Byzantine node. Let b_i be the number of Byzantine nodes in the i-th component in S_{Byz}, and note that $\sum_i b_i = b$. By Lemma 24.1, we know that in the absence of Byzantine nodes, the size of an attack component is at most $k_i \leq n/L$. Therefore, one Byzantine node can increase a component by at most n/L nodes plus itself. From this it follows that the size of an attack component i is bounded by

$$k_i \;\leq\; (b_i + 1) \cdot \frac{n}{L} + b_i, \quad \text{and} \quad l_i \;\leq\; (b_i + 1) \cdot \frac{n}{L}.$$

Using this relationship between b_i and the size of the attack component, we can bound $Cost_{inf}^{Byz}$ as

$$\begin{aligned} Cost_{inf}^{Byz} &= \frac{L}{n} \sum_{i \in S_{Byz}} l_i \cdot k_i \;\leq\; \frac{L}{n} \sum_{i \in S_{Byz}} \left[(b_i + 1) \cdot \frac{n}{L} \cdot \left((b_i + 1) \cdot \frac{n}{L} + b_i \right) \right] \\ &= \sum_{i \in S_{Byz}} \left[(b_i + 1)^2 \frac{n}{L} + b_i (b_i + 1) \right] \;<\; \sum_{i \in S_{Byz}} \left[(b_i + 1)^2 \left(\frac{n}{L} + 1 \right) \right] \\ &= \left(\frac{n}{L} + 1 \right) \cdot \sum_{i \in S_{Byz}} (b_i + 1)^2. \end{aligned}$$

Given the constraint that $b_i \geq 1$ for every b_i, and because $\sum_i b_i = b$, the above convex function assumes its maximum for a single positive $b_i = b$. Consequently,

$$Cost_{inf}^{Byz} \;\leq\; \left(\frac{n}{L}+1\right) \cdot \sum_{i \in S_{Byz}} (b_i+1)^2 \;\leq\; \left(\frac{n}{L}+1\right) \cdot (b+1)^2 \;\in\; O\left(\frac{b^2 n}{L}\right).$$

On the other hand, it clearly holds that at most every selfish node can be infected and hence, $Cost_{inf}^{\overline{Byz}} + Cost_{inf}^{Byz} \leq sL$. The proof is concluded by adding the upper bounds for $Cost_{inoc}$, $Cost_{inf}^{\overline{Byz}}$, and $Cost_{inf}^{Byz}$. □

Combining Lemmas 24.5 and 24.6 leads to the following theorem that captures the social cost in the virus inoculation game in the presence of b Byzantine players among selfish, oblivious nodes.

Theorem 24.7. *The social cost in a Byzantine Nash equilibrium with b Byzantine nodes in the oblivious model is $Cost_{BNE} \in \Theta(s + \frac{b^2 n}{L})$, for $b < \frac{L}{2} - 1$, and $Cost_{BNE} \in \Theta(sL)$, otherwise.*

Proof. In both cases, the lower bound follows from Lemma 24.5. As for the upper bound, note that for $b < \frac{L}{2} - 1$ and due to $L \leq n = s + b$, it holds that $b < \frac{s+b}{2}$ and therefore, $b < s$. Then, the term $s + \frac{b^2 n}{L}$ asymptotically cannot exceed the term sL and therefore, the claim follows. As for the second case, note that for $b \geq \frac{L}{2} - 1$, the term sL is asymptotically smaller or equal to $s + \frac{b^2 n}{L}$. □

Finally, we can derive tight bounds on the The Price of Byzantine Anarchy and the Price of Malice by bringing together the results of Theorems 24.2, 24.4, and 24.7.

Theorem 24.8. *In the virus inoculation game with b Byzantine nodes among selfish, oblivious nodes, the Price of Byzantine Anarchy and the Price of Malice are*

$$PoB(b) \in \Theta\left(\left(\frac{s}{L}\right)^{1/3}\left(1 + \frac{b^2}{L} + \frac{b^3}{sL}\right)\right) \quad \textit{and} \quad PoM(b) \in \Theta\left(1 + \frac{b^2}{L} + \frac{b^3}{sL}\right)$$

for $b < \frac{L}{2} - 1$. Otherwise, it holds that

$$PoB(b) \;\in\; \Theta\left(s^{1/3} L^{2/3}\right) \quad \textit{and} \quad PoM(b) \;\in\; \Theta\left(L\right).$$

Proof. Consider the case $b < \frac{L}{2} - 1$. For the Price of Byzantine Anarchy, we have $PoB(b) = \frac{Cost_{BNE}}{Cost_{OPT}} = \frac{\Theta(s + \frac{b^2(b+s)}{L})}{\Theta(s^{2/3}L^{1/3})} \in \Theta\left(\left(\frac{s}{L}\right)^{1/3} \cdot \left(1 + \frac{b^2}{L} + \frac{b^3}{sL}\right)\right)$. From this, the Price of Malice is computed as follows $PoM(b) = \frac{PoB(b)}{PoA} \in \Theta\left(1 + \frac{b^2}{L} + \frac{b^3}{sL}\right)$. The case $b \geq \frac{L}{2} - 1$ follows along the same lines by plugging in the corresponding expressions of Theorem 24.7. □

These results on the Price of Malice in the oblivious case support the intuition that in the absence of knowledge about the existence of Byzantine players, the quality of the global solution (i.e., the resulting social cost) deteriorates as the number of malicious players increases. In the next section, we show that the situation may change as soon as selfish players are *aware* of the existence of Byzantine players.

24.4.4 Non-oblivious Model

Having studied the oblivious model, we now turn our attention to the non-oblivious case in which selfish players know about the existence of Byzantine players. If selfish players knew about the exact locations of Byzantine nodes, they would be able to compute their optimal choice exactly. If they only know the *number of Byzantine nodes* in the system, however, the optimal strategy of a player becomes more complex. Specifically, it turns out that in this non-oblivious case, the "Fear Factor" may actually encourage players to act less selfishly and cooperate more. In fact, there may even be settings in which the existence of Byzantine players actually helps to improve the global social cost, rendering the Price of Malice to become less than 1.

Observe that in the non-oblivious case, every selfish node inoculates if $b \geq \frac{n}{L}$, implying a social cost of s. If $b < \frac{n}{L}$, the resulting social costs are bounded by the following lemma.

Lemma 24.9. *For $b < \frac{n}{2L}$, the social cost in a Byzantine Nash equilibrium in case of non-oblivious, risk-averse players with b Byzantine nodes is at least*

$$Cost_{BNE} \;\geq\; \frac{s}{2} + \frac{bL}{4}.$$

For all values of b, it holds that $Cost_{BNE} \geq \frac{s}{2}$.

Proof. We start with the more interesting case $b < \frac{n}{2L}$. Consider a grid with L columns each containing n/L nodes. All nodes in columns $2i+1$ for $i = 0, 1, \ldots, \frac{L}{2} - 1$ and all nodes in rows $j \cdot \frac{n/L-b}{b+1}$ for $j = 1, 2, \ldots, b$ are inoculated. That is, as illustrated in Figure 24.2, each component of insecure selfish nodes is of size $\frac{n/L-b}{b+1}$.

First, we show that this configuration constitutes a Byzantine Nash equilibrium in the risk-averse, non-oblivious case with b Byzantine nodes. Consider an *insecure node* in some column i. If all b secure nodes in this column are Byzantine, the size of the resulting attack component is $(n/L - b)/(b + 1) \cdot (b + 1) + b = n/L$. Hence, i's perceived infection cost is

$$\widehat{cost}_i \;=\; L \cdot \frac{(n/L - b)/(b+1) \cdot (b+1) + b}{n} \;=\; 1,$$

which equals the cost of inoculation. Next, consider an *inoculated selfish node* i and distinguish two cases. In the first case, i separates two components consisting of insecure selfish players and a change of i's strategy would merge

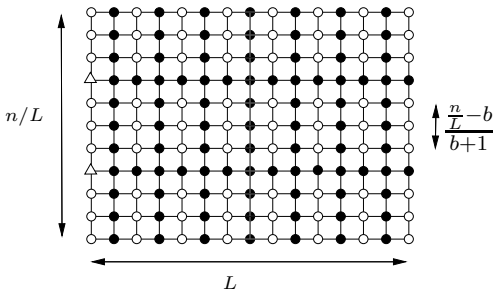

Figure 24.2: Example with large social cost for the non-oblivious, risk-averse model. White nodes are insecure, black nodes are secure, and white triangles denote Byzantine nodes.

two components of size $(n/L-b)/(b+1)$ into a single connected component of insecure selfish nodes. Every Byzantine node can connect another component of size $(n/L-b)/(b+1)$ (and itself) to the component containing i. Therefore, the size of the resulting attack component can be as large as $\left(2 \cdot \frac{\frac{n}{L}-b}{b+1} + 1\right) + \left(b \cdot \frac{\frac{n}{L}-b}{b+1} + b\right) = \frac{b+2}{b+1}\left(\frac{n}{L} - b\right) + b + 1 > \frac{n}{L} + \frac{1}{b+1}$. The perceived cost of i without inoculation is therefore

$$\widehat{cost}_i > L \cdot \frac{\frac{n}{L} + \frac{1}{b+1}}{n} = 1 + \frac{L}{n(b+1)} > 1.$$

In the second case, we consider a "crossing" node i that is located in the crossing of a secure row and column. Consider the column to the right (or to the left) of i. If all inoculated nodes in this column are Byzantine, the entire column plus node i becomes one large attack component. Hence, the perceived cost of i is

$$\widehat{cost}_i > L \cdot \frac{\frac{n}{L} + 1}{n} > 1.$$

In other words, no selfish node has an incentive to change its strategy and the situation in Figure 24.2 constitutes a Byzantine Nash equilibrium. In the sequel, we lower bound the social cost of this equilibrium under the assumption that all b Byzantine nodes are in column 1. Note that our construction guarantees that this is always possible if $b < \frac{n}{2L}$.

We start with the sum of the infection costs $Cost^0_{inf}$ of insecure nodes in column 0. The number of insecure, selfish nodes in this component is $\frac{n}{L} - b$. Hence, the expected sum of infection costs is

$$Cost^0_{inf} = \left(\frac{n}{L} - b\right) \cdot \frac{\frac{n}{L} - b + b}{n} \cdot L = \frac{n}{L} - b.$$

Let μ be the number of insecure nodes in columns $3, 5$, etc. The sum of the infection costs $Cost^r_{inf}$ of the remaining attack components (each being of size $\frac{n/L-b}{b+1}$) is

$$Cost^r_{inf} \;=\; \mu \cdot \frac{\frac{n}{L}-b}{n(b+1)} \cdot L \;>\; \mu \cdot \left(\frac{1}{b+1} - \frac{L}{n}\right).$$

Because the number of insecure nodes in these small attack components is $\mu = \frac{L-1}{2} \cdot \left(\frac{n}{L} - b\right)$, it follows that

$$\begin{aligned} Cost^r_{inf} &> \frac{L-1}{2} \cdot \left(\frac{n}{L} - b\right) \cdot \left(\frac{1}{b+1} - \frac{L}{n}\right) \\ &> \frac{1}{2(b+1)} \left(n - \frac{n}{L} - bL + b\right) - \frac{L}{2}. \end{aligned}$$

Finally, we also need to calculate the total inoculation cost of this topology. Clearly, all $s/2$ nodes in even columns are secure. (Recall that column and row indices start with 0.) Furthermore, b nodes in each odd column (except for the first column) are also inoculated. Hence, the total inoculation $Cost_{inoc}$ cost becomes

$$Cost_{inoc} \;=\; \frac{s}{2} + \frac{bL}{2} - b \;=\; \frac{s}{2} + b\left(\frac{L}{2} - 1\right).$$

Adding up all costs, the social cost of the Byzantine Nash equilibrium is

$$\begin{aligned} Cost_{BNE}(b) &\geq \frac{s}{2} + b\left(\frac{L}{2} - 1\right) + \frac{n}{L} - b + \frac{1}{2(b+1)}\left(n - \frac{n}{L} - bL + b\right) - \frac{L}{2} \\ &\geq \frac{s}{2} + \frac{bL}{4} \end{aligned}$$

for $b \leq \frac{n}{2L}$ and $b \geq 3$.

Finally, note that if $b \geq \frac{n}{2L}$, at least half of the selfish nodes inoculate and hence, $Cost_{BNE}(b) \geq s/2$. □

With this lower bound on the social cost of a Byzantine Nash equilibrium, we can now derive the Price of Byzantine Anarchy as well as the Price of Malice for the non-oblivious, risk-averse model.

Theorem 24.10. *In the non-oblivious, risk-averse model with b Byzantine nodes, the Price of Byzantine Anarchy is at least*

$$PoB(b) \;\geq\; \frac{1}{8}\left(\left(\frac{s}{L}\right)^{1/3} + \frac{b}{2}\left(\frac{L}{s}\right)^{2/3}\right)$$

for $b < \frac{n}{2L}$. For all b, it holds that $PoB(b) \geq \frac{1}{8}(\frac{s}{L})^{1/3}$.

Proof. Lemma 24.9 gives us a lower bound on the social cost of a Byzantine Nash equilibrium in the non-oblivious, risk-averse model with b malicious nodes. On the other hand, we have seen in Lemma 24.2, that the optimal social cost is at most $4s^{2/3}L^{1/3}$. Hence,

$$PoB(b) \quad \geq \quad \frac{\frac{s}{2}+\frac{bL}{4}}{4s^{2/3}L^{1/3}} \;=\; \frac{1}{8}\left(\frac{s^{1/3}}{L^{1/3}}+\frac{bL^{2/3}}{2s^{2/3}}\right).$$

The second lower bound follows analogously. □

Theorem 24.11. *In the non-oblivious, risk-averse model with b Byzantine nodes, the Price of Malice is*

$$PoM(b) \;\geq\; \frac{\sqrt{\pi}}{48}\left(1+\frac{bL}{2s}\right)$$

for $b < \frac{n}{2L}$. *For all b, it holds that* $PoM(b) \geq \frac{\sqrt{\pi}}{48}$.

Proof. In order to derive the Price of Malice, we can apply our bound from Theorem 24.10 and the upper bound on the Price of Anarchy established in Theorem 24.4. Specifically,

$$PoM(b) \;=\; \frac{PoB(b)}{PoA} \;\geq\; \frac{\frac{1}{8}\left(\left(\frac{s}{L}\right)^{1/3}+\frac{b}{2}\left(\frac{L}{s}\right)^{2/3}\right)}{\frac{6s^{1/3}}{\sqrt{\pi}\cdot L^{1/3}}}.$$

The theorem then follows from arithmetic simplifications. Again, the second lower bound follows in an analogous way. □

Discussion: From a technical point of view, this result shows that the Price of Malice may potentially be less than 1 in the non-oblivious model of the virus inoculation game. Intuitively, it is clear that in the presence of Byzantine players, nodes may be more willing to pay for inoculation. However, it is interesting that the selfish players' awareness of the existence of malicious Byzantine players may lead to an improvement of the overall system behavior, i.e., the *social welfare*. Specifically, the existence (or even the threat!) of malicious Byzantine players can render it worthwhile for nodes to enhance their cooperation to an extent which actually *improves* overall system performance as compared to a purely selfish system.

This highlights the existence of a *Fear Factor*, which describes the gain of the overall social efficiency in a system if selfish players are afraid of malicious, Byzantine individuals among them. This Fear Factor is determined by the ratio between the social cost of the worst Byzantine Nash equilibrium and the worst (regular) Nash equilibrium. Technically, we can define the Fear Factor Ψ as the inverse of the Price of Malice, i.e.,

$$\Psi(b) \;:=\; \frac{1}{PoM(b)}.$$

In other words, the Fear Factor Ψ quantifies how much the threat of a common enemy can unite selfish individuals, and to what degree the global social performance is improved.

In the virus inoculation game, the Fear Factor may be both negative and positive. What is interesting to note, however, is that it cannot be arbitrarily large, regardless of the number of Byzantine players b in the system. Instead, the Price of Malice can never drop below the constant $\frac{\sqrt{\pi}}{48}$ and hence, the Fear Factor is upper-bounded by $\Psi \leq \frac{48}{\sqrt{\pi}}$. That is, the social welfare or efficiency gained due to the Fear Factor cannot exceed a factor of $\Psi \leq \frac{48}{\sqrt{\pi}}$.

The existence of a Fear Factor has been documented in various economic and social models. The novel aspect brought about in this chapter is that by enhancing a game theoretic framework with the notion of Byzantine agents from distributed computing and cryptography, a *system's Fear Factor Ψ can be analytically quantified.*

Chapter 25

Conclusions and Outlook

The advent of the Internet and peer-to-peer systems has brought the notion of *selfishness* to the focus of theoretical computer science. As a result, the introduction of micro-economic models in computer science has led to fascinating insights into the reality of today's distributed systems such as the Internet. Over the last years, many aspects of distributed systems have been studied from a game-theoretic point of view. A particularly exciting question concerns the so-called *Price of Anarchy*: How much better would the social welfare be if selfish players collaborated instead of seeking to maximize their own benefit?

In Part IV of the thesis, we focused on two specific settings related to networking in which selfishness plays a crucial role. First, Chapter 23 has dealt with the example of locality-aware peer-to-peer systems in which the selfish nature of the participating peers may lead to inefficient and even instable topologies. In this game as well as in network creation games in general, there remain numerous open problems and directions for future work, many of which have already been pointed out in the original network creation paper [83]. For instance, it would be interesting to incorporate the notion of *congestion* into the framework or study games in which nodes arrive one-by-one and the network is developed in stages, each stage constituting an equilibrium.

One possible improvement that is specific to the peer-to-peer game studied in Chapter 23 is related to the node's *cost function*. It particular, it can be argued that minimizing the sum of the node's stretches may not be the most natural cost function. Instead, nodes may be more interested to minimize their *total latency* or their *maximum stretch*. Moreover, it may actually not be necessary for a node to be connected to every other node in the network. The motivation for studying the specific cost function defined in Chapter 23 is twofold. First, this cost function—while possibly not being the most realistic or intuitive one—does capture the classic peer-to-peer trade-off between latency on the one hand and maintenance overhead on the other hand. And secondly, the definition of the costs gives raise to an algorithmic

problem with manageable complexity and interesting results. This second reason is important because many other natural cost functions turn out to be analytically intricate or even intractable. Nonetheless, it will certainly be interesting to study versions of the locality game with different cost functions.

What happens if not every participant of a system is benevolent or (at least) acts according to its selfish interests? Chapter 24 advocates the study of distributed, potentially economic or social systems consisting of interacting players which can be selfish or malicious. Using a game-theoretic model that incorporates Byzantine players, we have derived bounds on the *Price of Malice* in oblivious and non-oblivious systems. Moreover, these results have lead to a quantification of the *Fear Factor*, which is the *gain* in system efficiency arising from the increased willingness of selfish individuals to cooperate in the view of malicious players.

The technical part of Chapter 24 focuses on a simple virus inoculation game on a restricted family of graphs. In this sense, these results may only scratch the surface and, generally, studying game theoretic settings with both selfish and Byzantine participants appears to open a multiplicity of algorithmically interesting questions and problems. For example: What is the Price of Malice in a virus inoculation game on a small-world graph? What is the Price of Malice of other games? It seems, for instance, that in certain routing games, a single node can attract a lot of traffic by announcing short distances to all other nodes resulting in a large Price of Malice. In congestions games, on the other hand, the impact of Byzantine players may be much smaller. Another direction for future work is to study the *impact of knowledge* on the resulting Fear Factor in non-oblivious models. Specifically, one could assume that players are not only aware of the existence of Byzantine players, but also of their approximate whereabouts or their statistical distribution. Intuitively, such additional knowledge should decrease the selfish players' incentive for collaboration and thus lower the Fear Factor.

Finally—if we are willing to let our minds wander even more freely at the conclusion of this thesis—modeling and studying the notions of *Price of Malice* and *Fear Factor* may lead to new insights in areas beyond those typically found in computer science and networking. Potentially, such game theoretic frameworks could provide tools for analytically capturing socio-economic artefacts arising in entirely different fields, including for example economics or sociology.

Bibliography

[1] I. Abraham, A. Badola, D. Bickson, D. Malkhi, S. Maloo, and S. Ron. Practical Locality-Awareness for Large Scale Information Sharing. In *Proc. of the 4^{th} Intl. Workshop on Peer-to-Peer Systems (IPTPS)*, pages 173–181, 2005.

[2] I. Abraham, D. Dolev, and D. Malkhi. LLS: A Locality Aware Location Service for Mobile Ad Hoc Networks. In *Proc. of 2nd Joint Workshop on Foundations of Mobile Computing (DIALM-POMC)*, pages 75–84, 2004.

[3] I. Abraham, C. Gavoille, A. V. Goldberg, and D. Malkhi. Routing in Networks with Low Doubling Dimension. In *Proceedings of the 26^{nd} International Conference on Distributed Computing Systems (ICDCS)*, 2006.

[4] I. Abraham, C. Gavoille, and D. Malkhi. Routing with Improved Communication-Space Trade-Off. In *Proc. of the 18^{th} Annual Conference on Distributed Computing (DISC)*, pages 305–319, 2004.

[5] I. Abraham, D. Malkhi, and O. Dobzinski. LAND: Stretch $(1 + e)$ Locality Aware Networks for DHTs. In *Proc. of the 15^{th} ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 550–559, 2004.

[6] E. Adar and B. Huberman. Free Riding on Gnutella. *First Monday*, 5(10), 2000.

[7] A. Aiyer, L. Alvisi, A. Clement, M. Dahlin, J.-P. Martin, and C. Porth. BAR Fault Tolerance for Cooperative Services. In *Proc. of the 20^{th} ACM Symposium on Operating Systems Principles (SOSP)*, pages 45–58, 2005.

[8] S. Albers, S. Eilts, E. Even-Dar, Y. Mansour, and L. Roditty. On Nash Equilibria for a Network Creation Game. In *Proc. of the 17^{th} ACM Symposium on Discrete Algorithms (SODA)*, Miami, USA, 2006.

[9] N. Alon, L. Babai, and A. Itai. A Fast and Simple Randomized Parallel Algorithm for the Maximal Independent Set Problem. *Journal of Algorithms*, 7(4):567–583, 1986.

[10] N. Alon, A. Bar-Noy, N. Linial, and D. Peleg. A Lower Bound for Radio Broadcast. *Journal of Computer and System Sciences*, 43:290–298, 1991.

[11] K. Alzoubi, P.-J. Wan, and O. Frieder. Message-Optimal Connected Dominating Sets in Mobile Ad Hoc Networks. In *Proc. of the 3^{rd} ACM Int. Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, pages 157–164, 2002.

[12] C. Ambühl, A. E. F. Clementi, M. D. Ianni, N. Lev-Tov, A. Monti, D. Peleg, G. Rossi, and R. Silvestri. Efficient Algorithms for Low-Energy Bounded-Hop Broadcast in Ad-Hoc Wireless Networks. In *Proc. of the 21^{th} Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 418–427, 2004.

[13] N. Andrade, M. Mowbray, A. Lima, G. Wagner, and M. Ripeanu. Influences on Cooperation in BitTorrent Communities. In *Proc. of the 2005 ACM SIGCOMM on Economics of Peer-to-Peer Systems*, pages 111–115, 2005.

[14] E. Anshelevich, A. Dasgupta, J. Kleinberg, E. Tardos, T. Wexler, and T. Roughgarden. The Price of Stability for Network Design with Fair Cost Allocation. In *Proc. of the 45^{th} Symposium on Foundations of Computer Science (FOCS)*, pages 295–304, 2004.

[15] J. Aspnes, K. Chang, and A. Yampolskiy. Inoculation Strategies for Victims of Viruses and the Sum-of-Squares Partition Problem. In *Proc. of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 43–52, 2005.

[16] H. Attiya and J. Welch. *Distributed Computing: Fundamentals, Simulations and Advanced Topics*. McGraw-Hill, 1998.

[17] B. Awerbuch. Complexity of Network Synchronization. *Journal of the ACM*, 32(4):804–823, 1985.

[18] B. Awerbuch. Optimal Distributed Algorithms for Minimum Weight Spanning Tree, Counting, Leader Election and Related Problems. In *Proc. of the 19^{th} ACM Symposium on Theory of Computing (STOC)*, pages 230–240, 1987.

[19] B. Awerbuch, B. Berger, L. Cowen, and D. Peleg. Low-Diameter Graph Decomposition is in NC. *Random Structures and Algorithms*, 5(3):441–452, 1994.

[20] B. Awerbuch, B. Berger, L. Cowen, and D. Peleg. Fast Network Decompositions and Covers. *Journal of Parallel and Distributed Computing*, 39(2):105–114, 1996.

[21] B. Awerbuch, A. V. Goldberg, M. Luby, and S. A. Plotkin. Network Decomposition and Locality in Distributed Computation. In *Proc. of the 30^{th} Symp. on Foundations of Computer Science (FOCS)*, pages 364–369, 1989.

[22] B. Awerbuch and D. Peleg. Network Synchronization with Polylogarithmic Overhead. In *Proc. of the 31^{th} Symposium on Foundations of Computer Science (FOCS)*, pages 514–522, 1990.

[23] B. Awerbuch and D. Peleg. Sparse Partitions. In *Proc. of the 31^{th} Symposium on Foundations of Computer Science (FOCS)*, pages 503–513, 1990.

[24] B. Awerbuch and D. Peleg. Routing with Polynomial Communication-Space Trade-Off. *SIAM Journal on Discrete Mathematics*, 5:151–162, 1992.

[25] B. Baker. Approximation Algorithms for NP-complete problems on Planar Graphs. *Journal of the ACM*, 41(1):153–180, 1994.

[26] M. L. Balinski. On Finding Integer Solutions to Linear Programs. In *Proc. of the IBM Scientific Computing Symposium on Combinatorial Problems*, pages 225–248, 1966.

[27] S. Banerjee and A. Misra. Minimum Energy Paths for Reliable Communication in Multi-Hop Wireless Networks. In *Proceedings of the 3^{rd} ACM International Symposium on Mobile Ad Hoc Networking & Computing (MOBIHOC)*, pages 146–156, 2002.

[28] J. Bar-Ilan, G. Kortsarz, and D. Peleg. How to Allocate Network Centers. *Journal of Algorithms*, 15(3):385–415, 1993.

[29] J. Bar-Ilan, G. Kortsarz, and D. Peleg. Generalized Submodular Cover Problems and Applications. *Theoretical Computer Science*, 250:179–200, 2001.

[30] R. Bar-Yehuda, O. Goldreich, and A. Itai. On the Time-Complexity of Broadcast in Radio Networks: An Exponential Gap Between Determinism and Randomization. In *Proceedings of the 6^{th} ACM Symposium on Principles of Distributed Computing (PODC)*, pages 98–108, 1987.

[31] Y. Bartal, J. W. Byers, and D. Raz. Global Optimization Using Local Information with Applications to Flow Control. In *Proc. of the 38^{th} Symposium on Foundations of Computer Science (FOCS)*, pages 303–312, 1997.

[32] Y. Bartal, J. W. Byers, and D. Raz. Global Optimization Using Local Information with Applications to Flow Control. In *Proc. of the 38^{th} Symposium on Foundations of Computer Science (FOCS)*, pages 303–312, 1997.

[33] A. Behzad and I. Rubin. On the Performance of Graph-based Scheduling Algorithms for Packet Radio Networks. In *Proc. of the IEEE Global Telecommunications Conference (GLOBECOM)*, pages 3432–3436, 2003.

[34] A. Behzad and I. Rubin. Impact of Power Control on the Performance of Ad Hoc Wireless Networks. In *Proc. of the 24^{th} Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2005.

[35] A. Behzad, I. Rubin, and A. Mojibi-Yazdi. Distributed Power Controlled Medium Access Control for Ad-Hoc Wireless Networks. In *Proc. of the 18^{th} IEEE Annual Workshop on Computer Communications (CCW)*, pages 47–53, 2003.

[36] P. Björklund, P. Värbrand, and D. Yuan. Resource Optimization of Spatial TDMA in Ad Hoc Radio Networks: A Column Generation Approach. In *Proc. of the 22^{th} Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2003.

[37] H. Breu and D. G. Kirkpatrick. Unit Disk Graph Recognition is NP-hard. *Computational Geometry. Theory and Applications*, 9(1-2):3–24, 1998.

[38] J. Bruck, J. Gao, and A. Jiang. Localization and Routing in Sensor Networks by Local Angle Information. In *Proc. of the 6^{th} International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, pages 181–192, 2005.

[39] M. Burkhart, P. von Rickenbach, R. Wattenhofer, and A. Zollinger. Does Topology Control Reduce Interference? In *Proceedings of the 5^{th} ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, pages 9–19, 2004.

[40] H. T.-H. Chan, A. Gupta, B. M. Maggs, and S. Zhou. On Hierarchical Routing in Doubling Metrics. In *Proc. of the 16^{th} Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 762–771, 2005.

[41] X. Cheng, X. Huang, D. Li, W. Wu, and D.-Z. Du. A Polynomial-Time Approximation Scheme for the Minimum-Connected Dominating Set in Ad Hoc Wireless Networks. *Networks*, 42(4):202–208, 2003.

[42] C.-F. Chiasserini and R. R. Rao. Routing Protocols to Maximize Battery Efficiency. In *IEEE Milcom 2000*, 2000.

[43] F. Chin and H. F. Ting. An Almost Linear Time and $O(n \log n + \epsilon)$ Messages Distributed Algorithm for Minimum-Weight Spanning Trees. In *Proc. of the 26^{th} ACM Symposium on Foundations of Computer Science (FOCS)*, pages 257–266, 1985.

[44] ChipCon AS. SmartRF CC1000 Datasheet (rev. 2.2). `http://www.chipcon.com/files/CC1000_Data_Sheet_2_2.pdf`.

[45] I. Chlamtac and S. Kutten. On Broadcasting in Radio Networks - Problem Analysis and Protocol Design. *IEEE Transactions on Communication*, 33:1240–1246, 1985.

[46] I. Chlamtac and O. Weinstein. The Wave Expansion Approach to Broadcasting in Multi-Hop Radio Networks. *IEEE Transactions on Communication*, 39, 1991.

[47] B. S. Chlebus. *Handbook of Randomized Computing*, chapter Randomized Communication in Radio Networks, pages 401–456. Kluwer Academic, 2001.

[48] B. S. Chlebus, L. Gąsieniec, A. Gibbons, A. Pelc, and W. Rytter. Deterministic Broadcasting in Unknown Radio Networks. *Distributed Computing*, 15(8), 2002.

[49] B. S. Chlebus, L. Gąsieniec, D. Kowalski, and T. Radzik. On the Wake-Up Problem in Radio Networks. In *Proc. of the 32^{rd} International Colloquium on Automata, Languages and Programming (ICALP)*, pages 347–359, 2005.

[50] B. S. Chlebus, L. Gąsieniec, A. Lingas, and A. Pagourtzis. Oblivious Gossiping in Ad-Hoc Radio Networks. In *Proc. of Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M)*, pages 44–51, 2001.

[51] B. S. Chlebus and D. Kowalski. A better Wake-Up in Radio Networks. In *Proc. of the 23^{rd} ACM Symposium on the Principles of Distributed Computing (PODC)*, pages 266–274, 2004.

[52] M. Chrobak, L. Gąsieniec, and D. Kowalski. The Wake-Up Problem in Multi-Hop Radio Networks. In *Proc. of the 15^{th} ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 992–1000, 2004.

[53] B.-G. Chun, K. Chaudhuri, H. Wee, M. Barreno, C. H. Papadimitriou, and J. Kubiatowicz. Selfish Caching in Distributed Systems: A Game-Theoretic Analysis. In *Proc. of the 23th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 21–30, 2004.

[54] V. Chvátal. A Greedy Heuristic for the Set-Covering Problem. *Mathematics of Operations Research*, 4(3), 1979.

[55] B. N. Clark, C. J. Colbourn, and D. S. Johnson. Unit Disk Graphs. *Discrete Mathematics*, 86(1-3):165–177, 1990.

[56] A. E. F. Clementi, P. Penna, and R. Silvestri. The Power Range Assignment Problem in Radio Networks on the Plane. In *Proc. of the 17^{th} Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 651–660, 2000.

[57] A. E. F. Clementi, P. Penna, and R. Silvestri. On the Power Assignment Problem in Radio Networks. *Mobile Networks and Applications*, 9(2):125–140, 2004.

[58] R. Cole and U. Vishkin. Deterministic Coin Tossing with Applications to Optimal Parallel List Ranking. *Information and Control*, 70(1):32–53, 1986.

[59] V. Conitzer and T. Sandholm. Complexity Results about Nash Equilibria. In *Proc. of the 18^{th} International Joint Conference on Artificial Intelligence (IJCAI)*, pages 765–771, 2003.

[60] J. Corbo and D. C. Parkes. The Price of Selfish Behavior in Bilateral Network Formation. In *Proc. of the 24^{th} ACM Symp. on Principles of Distributed Computing (PODC)*, pages 99–107, Las Vegas, Nevada, USA, 2005.

[61] G. Cornuejols, G. Nemhauser, and L. Wolsey. *Discrete Location Theory*, chapter The Uncapacitated Facility Location Problem, pages 119–171. Wiley, 1990.

[62] R. L. Cruz and A. V. Santhanam. Optimal Routing, Link Scheduling and Power Control in Multi-hop Wireless Networks. In *Proc. of the 22^{th} Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2003.

[63] A. Czumaj and W. Rytter. Broadcasting Algorithms in Radio Networks with Unknown Topology. In *Proc. of the 44^{th} Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 492–501. IEEE Computer Society, 2003.

[64] A. Czygrinow, M. Hańćkoviak, and E. Szymańska. A Fast Distributed Algorithm for Approximating the Maximum Matching. In *Proc. of the 12^{th} European Symposium on Algorithms (ESA)*, pages 252–263, 2004.

[65] F. Dai and J. Wu. An Extended Localized Algorithm for Connected Dominating Set Formation in Ad Hoc Wireless Networks. *IEEE Transactions on Parallel and Distributed Systems*, 15(10):902–920, 2004.

[66] M. Damian, S. Pandit, and S. Pemmaraju. Local Approximation Schemes for Topology Control. In *Proc. of the 25^{th} ACM Symposium on Principles of Distributed Computing (PODC)*, 2006.

[67] B. Deb and B. Nath. On the Node-Scheduling Approach to Topology Control in Ad Hoc Networks. In *Proc. of the 6^{nd} ACM Int. Symposium on Mobile Ad Hoc Networking & Computing (MOBIHOC)*, pages 14–26, 2005.

[68] E. D. Demaine, U. Feige, M. T. Hajiaghayi, and M. R. Salavatipour. Combination Can Be Hard: Approximability of the Unique Coverage Problem. In *Proc. of the 17^{th} ACM-SIAM Symposium on Discrete Algorithm (SODA)*, pages 162–171, 2006.

[69] I. Derbel and C. Gavoille. Fast Deterministic Distributed Algorithms for Sparse Spanners. In *Proc. of the 13th Colloquium on Structural Information and Communication Complexity (SIROCCO)*, 2006.

[70] D. Dolev. The Byzantine Generals Stike Again. *J. of Algorithms*, 3(1):14–30, 1982.

[71] S. Eidenbenz, V. Kumar, and S. Zust. Equilibria in Topology Control Games for Ad Hoc Networks. In *Proc. of the Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M)*, pages 2–11, 2003.

[72] F. Eisenbrand, S. Funke, N. Garg, and J. Könemann. A Combinatorial Algorithm for Computing a Maximum Independent Set in a t-perfect Graph. In *Proc. of the 14th Symposium on Discrete Algorithms (SODA)*, pages 517–522, 2003.

[73] T. ElBatt and A. Ephremides. Joint Scheduling and Power Control for Wireless Ad-hoc Networks. In *Proc. of the 21th Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2002.

[74] K. Eliaz. Fault Tolerant Implementation. *Review of Economic Studies*, 69:589–610, 2002.

[75] M. Elkin. A Faster Distributed Protocol for Constructing Minimum Spanning Tree. In *Proc. of the 15th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 352–361, 2004.

[76] M. Elkin. An Unconditional Lower Bound on the Hardness of Approximation of Distributed Minimum Spanning Tree Problem. In *Proc. of the 36th ACM Symposium on Theory of Computing (STOC)*, pages 331–340, 2004.

[77] M. Elkin. Distributed Approximation - A Survey. *ACM SIGACT News - Distributed Computing Column*, 35(4), 2004.

[78] M. Elkin and G. Kortsarz. Combinatorial Logarithmic Approximation Algorithm for Directed Telephone Broadcast Problem. In *Proc. of the 34th ACM Symposium on Theory of Computing (STOC)*, pages 438–447, 2002.

[79] M. Elkin and G. Kortsarz. Polylogarithmic Inapproximability of the Radio Broadcast Problem. In *Proc. of the 7th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 105–116, 2004.

[80] M. Elkin and G. Kortsarz. Improved Broadcast Schedule for Radio Networks. In *Proc. of the 16th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 222–231, 2005.

[81] A. Ephremides and T. Truong. Scheduling Broadcasts in Multihop Radio Networks. *IEEE Transactions on Communications*, 38:456–460, 1990.

[82] T. Erlebach, K. Jansen, and E. Seidel. Polynomial-Time Approximation Schemes for Geometric Graphs. In *Proc. of the 12^{th} ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 671–679, 2001.

[83] A. Fabrikant, A. Luthra, E. Maneva, C. H. Papadimitriou, and S. Shenker. On a Network Creation Game. In *Proc. of the 22^{nd} ACM Symposium on Principles of Distributed Computing (PODC)*, pages 347–351, 2003.

[84] A. Fabrikant, C. H. Papadimitriou, and K. Talwar. The Complexity of Pure Nash Equilibria. In *Proc. of the 36^{th} ACM Symposium on Theory of Computing (STOC)*, pages 604–612, 2004.

[85] M. Farach-Colton, R. J. Fernandes, and M. A. Mosteiro. Bootstrapping a Hop-Optimal Network in the Weak Sensor Model. In *Proc. of the 13^{th} European Symposium on Algorithms (ESA)*, pages 827–838, 2005.

[86] M. Farach-Colton, R. J. Fernandes, and M. A. Mosteiro. Lower Bounds for Clear Transmissions in Radio Networks. In *Proc. of the 7^{th} Latin American Symposium on Theoretical Informatics (LATIN)*, pages 447–454, 2006.

[87] U. Feige. A Threshold of $\ln n$ for Approximating Set Cover. *Journal of the ACM*, 45(4):634–652, 1998.

[88] U. Feige, M. M. Halldórsson, G. Kortsarz, and A. Srinivasan. Approximating the Domatic Number. *SIAM Journal on Computing*, 32(1):172–195, 2003.

[89] J. Feigenbaum, C. Papadimitriou, R. Sami, and S. Shenker. A BGP-based Mechanism for Lowest-Cost Routing. In *Proc. of the 21st Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 173–182, 2002.

[90] F. Fich and E. Ruppert. Hundreds of Impossibility Results for Distributed Computing. *Distributed Computing*, 16(2-3):121–163, 2003.

[91] I. Finocchi, A. Panconesi, and R. Silvestri. Experimental Analysis of Simple, Distributed Vertex Coloring Algorithms. In *Proceedings of the 13^{th} ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 606–615, 2002.

[92] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of Distributed Consensus With One Faulty Process. *Journal of the ACM*, 32(2):374–382, 1985.

[93] L. Fleischer. Approximating Fractional Multicommodity Flow Independent of the Number of Commodities. *SIAM Journal on Discrete Mathematics*, 13(4):505–520, 2000.

[94] L. Fleischer. A Fast Approximation Scheme for Fractional Covering Problems with Variable Upper Bounds. In *Proc. of the 15^{th} Symposium on Discrete Algorithms (SODA)*, 2004.

[95] S. Funke, A. Kesselman, U.Meyer, and M.Segal. A Simple Improved Distributed Algorithm for Minimum CDS in Unit Disk Graphs. In *Proc. of the 1^{st} Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, 2005.

[96] M. Fussen, R. Wattenhofer, and A. Zollinger. Interference Arises at the Receiver. In *Proc. of the International Conference on Wireless Networks, Communications, and Mobile Computing (WirelessCom)*, 2005.

[97] E. Gafni. Improvements in the Time Complexity of Two Message-Optimal Election Algorithms. In *Proc. of the 4^{th} Symposium on Principles of Distributed Computing (PODC)*, pages 175–185, 1985.

[98] R. G. Gallager, P. A. Humblet, and P. M. Spira. A Distributed Algorithm for Minimum-Weight Spanning Trees. *ACM Transactions on Programming Languages and Systems*, 5:66–77, 1983.

[99] R. Gandhi and S. Parthasarathy. Distributed Algorithms for Coloring and Connected Domination in Wireless Ad Hoc Networks. In *Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 447–459, 2004.

[100] J. Gao, L. Guibas, J. Hershberger, L. Zhang, and A. Zhu. Discrete Mobile Centers. In *Proc. of the 17^{th} Symposium on Computational Geometry (SCG)*, pages 188–196, 2001.

[101] J. Garay, S. Kutten, and D. Peleg. A Sub-Linear Time Distributed Algorithm for Minimum-Weight Spanning Trees. *SIAM Journal on Computing*, 27:302–316, 1998.

[102] M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[103] N. Garg and J. Koenemann. Faster and Simpler Algorithms for Multicommodity Flow and other Fractional Packing Problems. In *Proc. of the 39^{th} Annual Symposium on Foundations of Computer Science (FOCS)*, 1998.

[104] L. Gąsieniec, A. Pelc, and D. Peleg. The Wakeup Problem in Synchronous Broadcast Systems (Extended Abstract). In *Proc. of the 19^{th} ACM Symposium on Principles of Distributed Computing (PODC)*, pages 113–121, 2000.

[105] L. Gąsieniec, D. Peleg, and Q. Xin. Faster Communication in Known Topology Radio Networks. In *Proc. of the 24th ACM Symp. on Principles of Distributed Computing (PODC)*, pages 129–137, 2005.

[106] A. Goel and D. Estrin. Simultaneous Optimization for Concave Costs: Single Sink Aggregation or Single Source Buy-at-Bulk. In *Proc. of the 14th Symposium on Discrete Algorithms (SODA)*, 2003.

[107] M. X. Goemans and D. P. Williamson. A General Approximation Technique for Constrained Forest Problems. *SIAM Journal on Computing*, 24(2):296–317, 1995.

[108] A. Goldberg, S. Plotkin, and G. Shannon. Parallel Symmetry-Breaking in Sparse Graphs. *SIAM Journal on Discrete Mathematics*, 1(4):434–446, 1988.

[109] A. V. Goldberg and S. A. Plotkin. Parallel $(\Delta + 1)$-Coloring of Constant-degree Graphs. *Information Processing Letters*, 25:241–245, 1987.

[110] D. A. Grable and A. Panconesi. Fast Distributed Algorithms for Brooks-Vizing Colourings. In *Proc. of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 473–480, 1998.

[111] F. Grandoni, J. Krönemann, A. Panconesi, and M. Sozio. Primal-Dual Based Distributed Algorithms for Vertex Cover with Semi-Hard Capacities. In *Proc. of the 24th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 118–125, 2005.

[112] J. Grönkvist. Assignment methods for Spatial Reuse TDMA. In *Proc. of the 1st ACM International Symposium on Mobile Ad Hoc Networking & Computing (MOBIHOC)*, 2000.

[113] J. Grönkvist. Interference-Based Scheduling in Spatial Reuse TDMA. Doctoral thesis, 2005.

[114] J. Grönkvist and A. Hansson. Comparison Between Graph-Based and Interference-Based STDMA Scheduling. In *Proc. of the 2nd ACM International Symposium on Mobile Ad Hoc Networking & Computing (MOBIHOC)*, pages 255–258, 2001.

[115] A. Gupta, R. Krauthgamer, and J. Lee. Bounded Geometries, Fractals, and Low-Distortion Embeddings. In *Proc. of the 44th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 534–543, 2003.

[116] P. Gupta and P. R. Kumar. Critical Power for Asymptotic Connectivity in Wireless Networks. *Stochastic Analysis, Control, Optimization and Applications: A Volume in Honor of W. H. Fleming (March 1998)*, pages 547–566, 1998.

[117] P. Gupta and P. R. Kumar. The Capacity of Wireless Networks. *IEEE Trans. Information Theory*, 46(2):388–404, 2000.

[118] M. T. Hajiaghayi, M. Mahdian, and V. S. Mirrokni. The Facility Location Problem with General Cost Functions. *Networks*, 42(1):42–47, 2003.

[119] M. Hańćkoviak, M. Karoński, and A. Panconesi. On the Distributed Complexity of Computing Maximal Matchings. In *Proc. of the 9^{th} ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 219–225, 1998.

[120] M. Hańćkoviak, M. Karoński, and A. Panconesi. A faster Distributed Algorithm for Computing Maximal Matchings Deterministically. In *Proc. of the 18^{th} ACM Symposium on Principles of Distributed Computing (PODC)*, pages 219–228, 1999.

[121] J. Heinonen. *Lectures on Analysis of Metric Spaces*. Springer-Verlag, 2001.

[122] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-Efficient Communication Protocol for Wireless Microsensor Networks. In *Proc. of the 33^{rd} Annual Hawaii International Conference on System Sciences*, pages 3005–3014, 2000.

[123] J. Hill and D. Culler. Mica: A Wireless Platform for Deeply Embedded Networks. *IEEE Micro*, 22(6):12–24, 2002.

[124] D. Hochbaum and W. Maass. Approximation Schemes for Covering and Packing Problems. *Journal of the ACM*, 32(1):130–136, 1985.

[125] D. S. Hochbaum. Heuristics for the Fixed Cost Median Problem. *Math. Programming*, 22:148–162, 1982.

[126] L. Hu. Topology Control for Multihop Packet Radio Networks. *IEEE Trans. on Communications*, 41(10), 1993.

[127] H. Hunt, M. Marathe, V. Radhakrishnan, S. Ravi, D. Rosenkrantz, and R. Stearns. NC-Approximation Schemes for NP- and PSPACE-Hard Problems for Geometric Graphs. *ALGORITHMS: Journal of Algorithms*, 26, 1998.

[128] P. Indyk. Explicit Constructions of Selectors and Related Combinatorial Structures, with Applications. In *Proc. of the 13^{rd} ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 697–704, 2002.

[129] A. Israeli and A. Itai. A Fast and Simple Randomized Parallel Algorithm for Maximal Matching. *Information Processing Letters*, 22:77–80, 1986.

[130] K. Jain, M. Mahdian, E. Markakis, A. Saberi, and V. V. Vazirani. Greedy Facility Location Algorithms analyzed using Dual Fitting with Factor-Revealing LP. *Journal of the ACM*, 50(6):795–824, 2003.

[131] K. Jain, J. Padhye, V. Padmanabhan, and L. Qiu. Impact of Interference on Multi-hop Wireless Network Performance. In *Proc. of the 9^{th} Annual International Conference on Mobile Computing and Networking (MOBICOM)*, 2003.

[132] K. Jain and V. V. Vazirani. Primal-Dual Approximation Algorithms for Metric Facility Location and k-Median Problems. In *Proc. of the 40^{th} Annual Symposium on Foundations of Computer Science (FOCS)*, 1999.

[133] L. Jia, G. Lin, G. Noubir, R. Rajaraman, and R. Sundaram. Universal Approximations for TSP, Steiner Tree, and Set Cover. In *Proc. of the 37^{th} Annual ACM Symposium on Theory of Computing (STOC)*, pages 386–395, 2005.

[134] L. Jia, R. Rajaraman, and R. Suel. An Efficient Distributed Algorithm for Constructing Small Dominating Sets. In *Proc. of the 20^{th} ACM Symposium on Principles of Distributed Computing (PODC)*, pages 33–42, 2001.

[135] T. Johansson and L. Carr-Motyčkovà. Reducing Interference in Ad Hoc Networks through Topology Control. In *Proc. of the ACM Joint Workshop on Foundations of Mobile Computing (DIALM-POMC)*, pages 17–23, 2005.

[136] S. Jun and M. Ahamad. Incentives in BitTorrent Induce Free Riding. In *Proc. of the 2005 ACM SIGCOMM on Economics of Peer-to-Peer Systems*, pages 116–121, 2005.

[137] T. Jurdzinski, M. Kutylowski, and J. Zatopianski. Energy-Efficient Size Approximation of Radio Networks with No Collision Detection. In *Proc. of the 8^{th} Annual International Conference on Computing and Combinatorics (COCOON)*, pages 279–289, 2002.

[138] T. Jurdziński and G. Stachowiak. Probabilistic Algorithms for the Wakeup Problem in Single-Hop Radio Networks. In *Proc. of the 13^{th} Annual International Symposium on Algorithms and Computation (ISAAC)*, pages 535–549, 2002.

[139] D. R. Karger and M. Ruhl. Finding Nearest Neighbors in Growth-Restricted Metrics. In *Proc. of the 34^{th} Annual ACM Symposium on Theory of Computing (STOC)*, pages 741–750, 2002.

[140] R. M. Karp. Reducibility Among Combinatorial Problems. In *Proc. of a Symposium on the Complexity of Computer Computations*, pages 85–103, 1972.

[141] S. Kolliopoulos and N. Young. Tight Approximation Results for General Covering Integer Programs. In *Proc. of the 42nd IEEE Symposium on Foundations of Computer Science (FOCS)*, 2001.

[142] C.-Y. Koo. Broadcast in Radio Networks Tolerating Byzantine Adversarial Behavior. In *Proc. of the 23rd ACM Symposium on the Principles of Distributed Computing (PODC)*, pages 275–282, 2004.

[143] Y. A. Korilis, A. A. Lazar, and A. Orda. Achieving Network Optima using Stackelberg Routing. *Transactions on Networking*, 1997.

[144] M. R. Korupolu, C. G. Plaxton, and R. Rajaraman. Analysis of a Local Search Heuristic for Facility Location Problems. In *Proc. of the 9th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1–10, 1998.

[145] E. Koutsoupias and C. Papadimitriou. Worst-Case Equilibria. In *Proc. of the 16th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 404–413, 1999.

[146] R. Krauthgamer and J. Lee. Navigating Nets: Simple Algorithms for Proximity Search. In *Proc. of 15th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2004.

[147] F. Kuhn. The Price of Locality: Exploring the Complexity of Distributed Coordination Primitives. Doctoral thesis, eth zurich, 2005.

[148] F. Kuhn, T. Moscibroda, and R. Wattenhofer. Unit Disk Graph Approximation. In *Proc. of Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M)*, pages 17–23, 2004.

[149] F. Kuhn, T. Moscibroda, and R. Wattenhofer. Fault-Tolerant Clustering in Ad Hoc and Sensor Networks. In *Proceedings of the 26nd International Conference on Distributed Computing Systems (ICDCS)*, 2006.

[150] F. Kuhn, T. Moscibroda, and R. Wattenhofer. The Price of Being Near-Sighted. In *Proc. of the 17th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2006.

[151] F. Kuhn and R. Wattenhofer. Constant-Time Distributed Dominating Set Approximation. In *Proc. of the 22nd Annual ACM Symp. on Principles of Distributed Computing (PODC)*, pages 25–32, 2003.

[152] F. Kuhn and R. Wattenhofer. On the Complexity of Distributed Graph Coloring. In *Proc. of the 25th ACM Symposium on Principles of Distributed Computing (PODC)*, 2006.

[153] V. S. A. Kumar and M. V. Marathe. Improved Results for Stackelberg Scheduling Strategies. In *Proc. of the 29rd International Colloquium on Automata, Languages and Programming (ICALP)*, pages 776–787, 2002.

[154] V. S. A. Kumar, M. V. Marathe, S. Parthasarathy, and A. Srinivasan. Algorithmic Aspects of Capacity in Wireless Networks. In *Proc. International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, pages 133–144, 2005.

[155] E. Kushilevitz and Y. Mansour. An $\Omega(D \log(N/D))$ Lower Bound for Broadcast in Radio Networks. *SIAM Journal on Computing*, 27:702–712, 1998.

[156] S. Kutten and D. Peleg. Fast Distributed Construction of Small k-Dominating Sets and Applications. *Journal of Algorithms*, 28:40–66, 1998.

[157] L. Lamport, R. Shostak, and M. Pease. The Byzantine Generals Problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

[158] F. Lazebnik and V. A. Ustimenko. Explicit Construction of Graphs with an Arbitrary Large Girth and of Large Size. *Discrete Applied Mathematics*, 60(1-3).

[159] F. Lazebnik, V. A. Ustimenko, and A. J. Woldar. A New Series of Dense Graphs of High Girth. *Bulletin of the American Mathematical Society (N.S.)*, 32(1):73–79, 1995.

[160] T. Leighton, B. Maggs, and S. Rao. Scheduling in O(Congestion+Dilation) Steps. *Combinatorica*, 14(2):167–180, 1994.

[161] N. Li, C.-J. Hou, and L. Sha. Design and Analysis of an MST-Based Topology Control Algorithm. In *Proc. of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2003.

[162] N. Li and J. Hou. Topology Control in Heterogenous Wireless Networks: Problems and Solutinos. In *Proc. of the 23rd Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2004.

[163] X.-Y. Li, G. Calinescu, and P.-J. Wan. Distributed Construction of Planar Spanner and Routing for Ad Hoc Wireless Networks. In *Proc. of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2002.

[164] X.-Y. Li, W.-Z. Song, and W. Wang. A Unified Energy Efficient Topology for Unicast and Broadcast. In *Proc. of the 11th International Conference on Mobile Computing and Networking (MOBICOM)*, pages 1–15, 2005.

[165] J.-H. Lin and J. S. Vitter. ϵ-Approximations with Minimum Packing Constraint Violation. In *Proc. of the 24th ACM Symposium on Theory of Computing (STOC)*, pages 771–782, 1992.

[166] N. Linial. Locality in Distributed Graph Algorithms. *SIAM Journal on Computing*, 21(1):193–201, 1992.

[167] N. Linial and M. Saks. Low Diameter Graph Decompositions. *Combinatorica*, 13(4):441–454, 1993.

[168] Z. Lotker, E. Pavlov, B. Patt-Shamir, and D. Peleg. MST Construction in $O(\log \log n)$ Communication Rounds. In *Proc. of the 15th Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 94–100, 2003.

[169] M. Luby. A Simple Parallel Algorithm for the Maximal Independent Set Problem. *SIAM Journal on Computing*, 15:1036–1053, 1986.

[170] M. Luby and N. Nisan. A Parallel Approximation Algorithm for Positive Linear Programming. In *Proc. of the 25th ACM Symposium on Theory of Computing (STOC)*, pages 448–457, 1993.

[171] C. Lund and M. Yannakakis. On the Hardness of Approximating Minimization Problems. *Journal of the ACM*, 41(5):960–981, 1994.

[172] D. Malkhi and M. Reiter. Byzantine Quorum Systems. *Journal of Distributed Computing*, 11(4):203–213, 1998.

[173] M. V. Marathe, H. Breu, H. B. Hunt III, S. S. Ravi, and D. J. Rosenkrantz. Simple Heuristics for Unit Disk Graphs. *Networks*, 25:59–68, 1995.

[174] M. J. McGlynn and S. A. Borbash. Birthday Protocols for Low Energy Deployment and Flexible Neighborhood Discovery in Ad Hoc Wireless Networks. In *Proc. of the 2nd ACM Int. Symposium on Mobile Ad Hoc Networking & Computing (MOBIHOC)*, 2001.

[175] F. Meyer auf der Heide, C. Schindelhauer, K. Volbert, and M. Grünewald. Energy, Congestion and Dilation in Radio Networks. In *Proc. of the 14th ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 230–237, 2002.

[176] M. J. Miller, C. Sengul, and I. Gupta. Exploring the Energy-Latency Trade-off for Broadcasts in Energy-Saving Sensor Networks. In *Proc. of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2005.

[177] K. Moaveni-Nejad and X.-Y. Li. Low-Interference Topology Control for Wireless Ad Hoc Networks. In *Proc. of the 2nd IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON)*, Santa Clara, California, USA, 2005.

[178] P. Monks, V. Bharghavan, and W. W. Hwu. A Power Controlled Multiple Access Protocol for Wireless Packet Networks. In *Proceedings of the 20^{th} Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 1567–1576, 2001.

[179] T. Moscibroda and R. Wattenhofer. Maximizing the Lifetime of Dominating Sets. In *Proceedings of the 5^{th} International Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks (WMAN)*, 2005.

[180] T. Moscibroda and R. Wattenhofer. Minimizing Interference in Ad Hoc and Sensor Networks. In *Proc. of the ACM Joint Workshop on Foundations of Mobile Computing (DIALM-POMC)*, 2005.

[181] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[182] K. Nakano and S. Olariu. Energy-Efficient Initialization Protocols for Single-Hop Radio Networks with no Collision Detection. *IEEE Transactions on Parallel and Distributed Systems*, 11(8), 2000.

[183] K. Nakano and S. Olariu. A Survey on Leader Election Protocols for Radio Networks. In *Proc. of the 6^{th} International Symposium on Parallel Architectures, Algorithms, and Networks (ISPAN)*, pages 71–78, 2002.

[184] M. Naor and L. Stockmeyer. What Can Be Computed Locally? *SIAM Journal on Computing*, 24(6):1259–1277, 1995.

[185] T. Nieberg and J. Hurink. A PTAS for the Minimum Dominating Set Problem in Unit Disk Graphs. In *Proc. of the 3^{rd} Workshop on Approximation and Online Algorithms (WAOA)*, 2005.

[186] T. Nieberg, J. Hurink, and W. Kern. A Robust PTAS for Maximum Independent Sets in Unit Disk Graphs. In *Proc. of the 30^{th} Workshop on Graph Theoretic Concepts in Computer Science (WG)*, pages 214–221, 2004.

[187] N. Nisan and A. Ronen. Algorithmic Mechanism Design. In *Proc. of the 31^{rd} ACM Symposium on Theory of Computing (STOC)*, pages 129–140, 1999.

[188] A. Panconesi and R. Rizzi. Some Simple Distributed Algorithms for Sparse Networks. *Distributed Computing*, 14(2):97–100, 2001.

[189] A. Panconesi and A. Srinivasan. Improved Distributed Algorithms for Coloring and Network Decomposition Problems. In *Proc. of the 24^{th} annual ACM symposium on Theory of computing (STOC)*, pages 581–592, 1992.

[190] C. H. Papadimitriou. Algorithms, Games, and the Internet. In *Proc. of the* 33^{rd} *ACM Symposium on Theory of Computing (STOC)*, pages 749–753, 2001.

[191] C. H. Papadimitriou. Computing Correlated Equilibria in Multi-Player Games. In *Proc. of the* 37^{th} *ACM Symposium on Theory of Computing (STOC)*, pages 49–56, 2005.

[192] C. H. Papadimitriou and M. Yannakakis. On the Value of Information in Distributed Decision Making. In *Proc. of the* 10^{th} *ACM Symposium on Principles of Distributed Computing (PODC)*, pages 61–64, 1991.

[193] C. H. Papadimitriou and M. Yannakakis. Linear Programming Without the Matrix. In *Proc. of the* 25^{th} *ACM Symposium on Theory of Computing (STOC)*, pages 121–129, 1993.

[194] D. Peleg. Sparse Graph Partitions. Technical Report CS89-01, The Weizmann Institute of Science, 1989.

[195] D. Peleg. Distance-Dependent Distributed Directories. *Information and Communication*, 103:270–298, 1993.

[196] D. Peleg. *Distributed Computing: A Locality-Sensitive Approach*. SIAM Monographs on Discrete Mathematics and Applications, 2000.

[197] D. Peleg and V. Rubinovich. A Near-Tight Lower Bound on the Time Complexity of Distributed Minimum-Weight Spanning Tree Construction. *SIAM Journal on Computing*, 30(5):1427–1442, 2001.

[198] S. Pemmaraju and I. Pirwani. Energy-Conservation in Wireless Sensor Networks via Domatic Partitions. In *Proc. of the* 7^{th} *International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, 2006.

[199] C. Plaxton, R. Rajaraman, and A. W. Richa. Accessing Nearby Copies of Replicated Objects in a Distributed Environment. In *Proc. of the* 9^{th} *ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 311–320, 1997.

[200] S. Plotkin, D. Shmoys, and E. Tardos. Fast Approximation Algorithms for Fractional Packing and Covering Problems. *Mathematics of Operations Research*, 20:257–301, 1995.

[201] J. Polastre, J. Hill, and D. Culler. Versatile Low Power Media Access for Wireless Sensor Networks. In *Proc. of the* 2^{nd} *Int. Conference on Embedded Networked Sensor Systems (SENSYS)*, pages 95–107, 2004.

[202] S. Rajagopalan and V. Vazirani. Primal-Dual RNC Approximation Algorithms for Set Cover and Covering Integer Programs. *SIAM Journal on Computing*, 28:525–540, 1998.

[203] V. Rajendran, K. Obraczka, and J. J. Garcia-Luna-Aceves. Energy-Efficient Collision-Free Medium Access Control for Wireless Sensor Networks. In *Proceedings of the 1^{st} International Conference on Embedded Networked Sensor Systems (SENSYS)*, pages 181–192, 2003.

[204] R. Ramanathan and R. Rosales-Hain. Topology Control of Multihop Wireless Networks Using Transmit Power Adjustment. In *Proceedings of the 19^{th} Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2000.

[205] S. Ramanathan and E. L. Lloyd. Scheduling Algorithms for Multi-Hhop Radio Networks. In *Proc. of the Conference on Communications Architectures & Protocols (SIGCOMM)*, pages 211–222, 1992.

[206] V. Rodoplu and T. H. Meng. Minimum Energy Mobile Wireless Networks. *IEEE J. Selected Areas in Communications*, 17(8), 1999.

[207] T. Roughgarden. Stackelberg Scheduling Strategies. In *Proc. of the 33^{rd} ACM Symposium on Theory of Computing (STOC)*, pages 104–113, 2001.

[208] T. Roughgarden. *Selfish Routing and the Price of Anarchy*. MIT Press, 2005.

[209] T. Roughgarden and E. Tardos. How Bad is Selfish Routing? *Journal of the ACM*, 49(2), 2002.

[210] A. Rowstron and P. Druschel. Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems. In *Proc. of the IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, 2001.

[211] A. Sen and M. L. Huson. A new Model for Scheduling Packet Radio Networks. In *Proc. of the 15^{th} Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 1116–1124, 1996.

[212] D. B. Shmoys, E. Tardos, and K. I. Aardal. Approximation Algorithms for Facility Location Problems. In *Proc. of the 29^{th} ACM Symposium on Theory of Computing (STOC)*, pages 265–274, 1997.

[213] R. Shostak, M. Pease, and L. Lamport. Reaching Agreement in the Presence of Faults. *J. of the ACM*, 27(2):228–234, 1980.

[214] S. Singh and C. S. Raghavendra. PAMAS - Power Aware Multi-Access Protocol with Signalling for Ad Hoc Networks. *SIGCOMM Comput. Commun. Rev.*, 28(3):5–26, 1998.

[215] A. Sinha and A. Chandrakasan. Dynamic Power Management in Wireless Sensor Networks. *IEEE Design and Test*, 18(2):62–74, 2001.

[216] P. Sinha, R. Sivakumar, and V. Bharghavan. Enhancing Ad Hoc Routing with Dynamic Virtual Infrastructures. In *Proc. of the 20^{th} Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 1763–1772, 2001.

[217] A. Slivkins. Distance Estimation and Object Location via Rings of Neighbors. In *Proc. of the 24^{th} ACM Symposium on Principles of Distributed Computing (PODC)*, pages 41–50, 2005.

[218] D. Son, B. Krishnamachari, and J. Heidemann. Experimental Analysis of Concurrent Packet Transmissions in Low-Power Wireless Networks. Technical report, Viterbi School of Engineering, University of Southern California, 2005.

[219] T. K. Srikant and S. Toueg. Simulating Authenticated Broadcasts to Derive Simple Fault-Tolerant Algorithms. *Journal of Distributed Computing*, 2(2):80–94, 1987.

[220] C. Swamy and D. B. Shmoys. Fault-Tolerant Facility Location. In *Proc. of the 14^{th} ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 735–736. Society for Industrial and Applied Mathematics, 2003.

[221] K. Talwar. Bypassing the Embedding: Approximation Schemes and Compact Representations for Low Dimensional Metrics. In *Proc. of 36th ACM Symposium on Theory of Computing (STOC)*, 2004.

[222] F. A. Tobagi and L. Kleinrock. Packet Switching in Radio Channels: Part II - The Hidden Terminal Problem in Carrier Sense Multiple Access and the Busy Tone Solution. COM-23(12):1417–1433, 1975.

[223] V. V. Vazirani. *Approximation Algorithms*. Springer, Berlin, 2001.

[224] P. von Rickenbach, S. Schmid, R. Wattenhofer, and A. Zollinger. A Robust Interference Model for Wireless Ad-Hoc Networks. In *Proc. of the 5^{th} Int. Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks (WMAN)*, Denver, Colorado, USA, 2005.

[225] P. von Rickenbach and R. Wattenhofer. Gathering Correlated Data in Sensor Networks. In *Proc. of the ACM Joint Workshop on Foundations of Mobile Computing (DIALM-POMC)*, 2004.

[226] H. von Stackelberg. *Marktform und Gleichgewicht*. Springer-Verlag, 1934.

[227] P. Wan, K. Alzoubi, and O. Frieder. Distributed Construction of Connected Dominating Set in Wireless Ad Hoc Networks. In *Proc. of the 21^{th} Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2002.

[228] P.-J. Wan, G. Calinescu, X.-Y. Li, and O. Frieder. Minimum-Energy Broadcast Routing in Static Ad Hoc Wireless Networks. In *Proc. of the 20^{th} Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2001.

[229] Y. Wang, W. Wang, and X.-Y. Li. Distributed Low-Cost Backbone Formation for Wireless Ad Hoc Networks. In *Proceedings of the 6^{th} ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, pages 2–13, 2005.

[230] M. Wattenhofer and R. Wattenhofer. Distributed Weighted Matching. In *Proc. of the 18^{th} Annual Conference on Distributed Computing (DISC)*, pages 335–348, 2004.

[231] R. Wattenhofer, L. Li, P. Bahl, and Y.-M. Wang. Distributed Topology Control for Power Efficient Operation in Multihop Wireless Ad Hoc Networks. In *Proc. of the 20^{th} Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2001.

[232] R. Wattenhofer and A. Zollinger. XTC: A Practical Topology Control Algorithm for Ad-Hoc Networks. In *Proc. of the 4^{th} Int. Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks (WMAN)*, Santa Fe, New Mexico, USA, 2004.

[233] J. L. Welch and N. Lynch. A New Fault-Tolerant for Clock-Synchronization. *Information and Communication*, 77:1–36, 1988.

[234] J. E. Wieselthier, G. D. Nguyen, and A. Ephremides. Energy-Efficient Broadcast and Multicast Trees in Wireless Networks. *Mobile Networks and Applications (MONET)*, 7(6):481–492, 2002.

[235] D. P. Williamson, M. X. Goemans, M. Mihail, and V. V. Vazirani. A Primal-Dual Approximation Algorithm for Generalized Steiner Network Problems. In *Proc. of the 25^{th} ACM Symposium on Theory of Computing (STOC)*, pages 708–717. ACM Press, 1993.

[236] B. Wong, A. Slivkins, and E. G. Sirer. Meridian: A Lightweight Network Location Service without Virtual Coordinates. In *Proc. of the Conference on Communications Architectures & Protocols (SIGCOMM)*, 2005.

[237] A. Woo and D.-E. Culler. A Transmission Control Scheme for Media Access in Sensor Networks. In *Proc. of the 7^{th} Annual International Conference on Mobile Computing and Networking (MOBIHOC)*, pages 221–235. ACM Press, 2001.

[238] J. Wu and H. Li. On Calculating Connected Dominating Set for Efficient Routing in Ad Hoc Wireless Networks. In *Proc. of the 3^{rd} Int. Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIALM)*, pages 7–14, 1999.

[239] M. Yang, Z. Zhang, X. Li, and Y. Dai. An Empirical Study of Free-Riding Behavior in the Maze P2P File Sharing System. In *Proc. of the 4^{th} Intl. Workshop on Peer-to-Peer Systems (IPTPS)*, 2005.

[240] W. Ye, J. Heidemann, and D. Estrin. An Energy-Efficient MAC protocol for Wireless Sensor Networks. In *Proc. of the 22^{th} Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 1567–1576, 2002.

[241] N. Young. Sequential and Parallel Algorithms for Mixed Packing and Covering. In *Proc. of the 42^{nd} IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 538–546, 2001.

[242] Y. Yu, B. Krishnamachari, and V. K. Prasanna. Energy-Latency Tradeoffs for Data Gathering in Wireless Sensor Networks. In *Proc. of the 23^{st} Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2004.

[243] B. Y. Zhao, L. Huang, J. Stribling, A. D. Joseph, and J. D. Kubiatowicz. Tapestry: A Resilient Global-scale Overlay for Service Deployment. *IEEE Journal on Selected Areas in Communications*, 22(1), 2004.

Curriculum Vitae

September 11, 1979	Born in Lucerne, Switzerland
1985–1999	Primary, secondary, and high schools in Malters and Reussbühl, Switzerland
1999–2004	Studies in computer science, ETH Zurich, Switzerland
April 2004	M.Sc. in computer science, ETH Zurich, Switzerland
2004–2006	Ph.D. student, research and teaching assistant, Distributed Computing Group, Prof. Roger Wattenhofer, ETH Zurich, Switzerland
July 2006	Ph.D. degree, Distributed Computing Group, ETH Zurich, Switzerland Advisor: Prof. Roger Wattenhofer Co-examiners: Prof. Christos H. Papadimitriou UC Berkeley, California, USA Prof. David Peleg Weizmann Institute of Science, Israel

Publications

The following lists all publications written during the two and a half years of my being Ph.D. student at ETH Zurich.

1. *When Selfish Meets Evil: Byzantine Players in a Virus Inoculation Game.* Thomas Moscibroda, Stefan Schmid, and Roger Wattenhofer. 25th ACM Symposium on the Principles of Distributed Computing (PODC), Denver, Colorado, USA, July 2006.

2. *On the Topologies Formed by Selfish Peers.* Thomas Moscibroda, Stefan Schmid, and Roger Wattenhofer. 25th ACM Symposium on the Principles of Distributed Computing (PODC), Denver, Colorado, USA, July 2006.

   Bibinfo: A preliminary version of the paper was also accepted for presentation at the 5th International Workshop on Peer-to-Peer Systems (IPTPS), Santa Barbara, California, USA, February 2006.

3. *Fault-Tolerant Clustering in Ad Hoc and Sensor Networks.* Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. 26th International Conference on Distributed Computing Systems (ICDCS), Lisbon, Portugal, July 2006.

4. *Topology Control Meets SINR: The Scheduling Complexity of Arbitrary Topologies.* Thomas Moscibroda, Roger Wattenhofer, and Aaron Zollinger. 7th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC), Florence, Italy, May 2006.

5. *The Complexity of Connectivity in Wireless Networks.* Thomas Moscibroda and Roger Wattenhofer. 25th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), Barcelona, Spain, April 2006.

6. *Analyzing the Energy-Latency Trade-off during the Deployment of Sensor Networks.* Thomas Moscibroda, Pascal von Rickenbach, and Roger Wattenhofer. 25th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), Barcelona, Spain, April 2006.

7. *The Price of Being Near-Sighted.* Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. 17th ACM-SIAM Symposium on Discrete Algorithms (SODA), Miami, Florida, USA, January 2006.

8. *Fast Deterministic Distributed Maximal Independent Set Computation on Growth-Bounded Graphs.* Fabian Kuhn, Thomas Moscibroda, Tim Nieberg, and Roger Wattenhofer. 19th International Symposium on Distributed Computing (DISC), Cracow, Poland, September 2005.

9. *Minimizing Interference in Ad Hoc and Sensor Networks.* Thomas Moscibroda and Roger Wattenhofer. 3rd ACM Joint Workshop on Foundations of Mobile Computing (DIALM-POMC), Cologne, Germany, September 2005.

10. *Local Approximation Schemes for Ad Hoc and Sensor Networks.* Fabian Kuhn, Thomas Moscibroda, Tim Nieberg, and Roger Wattenhofer. 3rd ACM Joint Workshop on Foundations of Mobile Computing (DIALM-POMC), Cologne, Germany, September 2005.

11. *Coloring Unstructured Radio Networks.* Thomas Moscibroda and Roger Wattenhofer. 17th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA), Las Vegas, Nevada, USA, July 2005.

12. *Maximal Independent Sets in Radio Networks.* Thomas Moscibroda and Roger Wattenhofer. 24th ACM Symposium on the Principles of Distributed Computing (PODC), Las Vegas, Nevada, USA, July 2005.

13. *On the Locality of Bounded Growth.* Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. 24th ACM Symposium on the Principles of Distributed Computing (PODC), Las Vegas, Nevada, USA, July 2005.

14. *Facility Location: Distributed Approximation.* Thomas Moscibroda and Roger Wattenhofer. 24th ACM Symposium on the Principles of Distributed Computing (PODC), Las Vegas, Nevada, USA, July 2005.

15. *Maximizing the Lifetime of Dominating Sets.* Thomas Moscibroda and Roger Wattenhofer. 5th International Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks (WMAN), Denver, Colorado, April 2005.

16. *Efficient Computation of Maximal Independent Sets in Unstructured Multi-Hop Radio Networks.* Thomas Moscibroda and Roger Wattenhofer. 1st IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS), Fort Lauderdale, Florida, USA, October 2004.

17. *Unit Disk Graph Approximation.* Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. ACM Joint Workshop on Foundations of Mobile Computing (DIALM-POMC), Philadelphia, Pennsylvania, USA, October 2004.

18. *Virtual Coordinates for Ad hoc and Sensor Networks.* Thomas Moscibroda, Regina O'Dell, Mirjam Wattenhofer, and Roger Wattenhofer. ACM Joint Workshop on Foundations of Mobile Computing (DIALM-POMC), Philadelphia, Pennsylvania, USA, October 2004.

19. *Initializing Newly Deployed Ad Hoc and Sensor Networks.* Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer
10th Annual International Conference on Mobile Computing and Networking (MOBICOM), Philadelphia, USA, September 2004.

20. *Radio Network Clustering from Scratch.* Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. 12nd Annual European Symposium on Algorithms (ESA), Bergen, Norway, September 2004.

21. *Brief Announcement: Efficient Clustering in Unstructured Radio Networks.* Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. 23rd ACM Symposium on the Principles of Distributed Computing (PODC), St. John's, Newfoundland, Canada, July 2004.

22. *What Cannot Be Computed Locally!* Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. 23rd ACM Symposium on the Principles of Distributed Computing (PODC), St. John's, Newfoundland, Canada, July 2004.