DISS. ETH NO. 16213

The Price of Locality:

Exploring the Complexity of Distributed Coordination Primitives

A dissertation submitted to the

SWISS FEDERAL INSTITUTE OF TECHNOLOGY ZURICH

for the degree of

Doctor of Sciences

presented by

FABIAN KUHN

Dipl. Inf.-Ing., ETH Zürich

born 30.07.1976

citizen of

Waltenschwil AG

accepted on the recommendation of

Prof. Dr. Roger Wattenhofer, examiner
Prof. Dr. Nathan Linial, co-examiner
Prof. Dr. Friedhelm Meyer auf der Heide, co-examiner

2005

Abstract

In a distributed computation, many autonomous processors intend to jointly come up with a solution for a given problem. It is usually assumed that the processors are located at different sites and are connected to each other by a network. Coordination between different processors (also called network nodes) is done by exchanging messages. If the input for a given problem is distributed among the nodes of the network, nodes need to communicate in order to learn about each other's input.

In k units of time, it is only possible to send information to nodes at distance at most k in the network. If the diameter of a given network is large, no node can learn about the whole network. In a few time units, a node can only gather data from nearby nodes. Thus, all nodes have to base their computations on partial information. Nevertheless, all network nodes together have to reach a common solution for the problem at hand. Achieving a global goal based on local information is therefore one of the most fundamental problems in the area of distributed computing.

This thesis considers the complexity of different important distributed coordination tasks. In particular, three major problems are discussed. First, we establish nearly tight upper and lower bounds on the possible trade-offs between running time and quality of the solution for a class of problems called covering and packing problems. Prominent network-related members of this problem class are for example minimum dominating set and maximum matching. Second, we consider protocols for coloring the nodes of the network. We analyze the potential of the simplest coloring algorithms where each node decides on a color based on the colors or identifiers of its direct neighbors only. Third, we study the distributed complexity of problems on a special kind of network topologies occurring in the context of wireless networks such as ad hoc or sensor networks.

Zusammenfassung

Bei einer verteilten Berechnung beabsichtigen mehrere autonome Prozessoren, gemeinsam ein Problem zu lösen. Normalerweise wird angenommen, dass sich die Prozessoren an verschiedenen Orten befinden und untereinander durch ein Netzwerk verbunden sind. Die Koordination zwischen verschiedenen Prozessoren, in diesem Fall auch Netzwerkknoten genannt, geschieht durch das Austauschen von Nachrichten. Falls die Eingabe fuer ein gegebenes Problem auf verschiedene Knoten verteilt ist, müssen die Knoten miteinander kommunizieren, um andere Teile der Eingabe zu erfahren.

In k Zeiteinheiten können Informationen nur zu Knoten gesendet werden, welche über höchstens $k-1$ zusätzliche Knoten erreicht werden können. Falls der Durchmesser eines Netzwerks gross ist, kann kein Knoten das ganze Netzwerk kennenlernen. In ein paar Zeiteinheiten kann ein Knoten nur Daten von naheliegenden Knoten einsammeln. Trotzdem müssen alle Knoten zusammen eine gemeinsame Lösung für das gegebene Problem finden. Basierend auf lokaler Information ein gemeinsames, globales Ziel zu erreichen ist deshalb eines der wichtigsten Probleme im Bereich der verteilten Algorithmen.

Diese Dissertation analysiert die Komplexität von verschiedenen wichtigen, verteilten Koordinationsaufgaben. Es werden drei grosse Probleme diskutiert. Als erstes betrachten wir sogenannte "Covering"- und "Packing"-Probleme. Wichtige Netzwerk-relevante Probleme dieser Klasse sind zum Beispiel "Minimum Dominating Set" und "Maximum Matching". Es werden fast übereinstimmende untere und obere Schranken für die Beziehung zwischen Laufzeit und Qualität der Lösung hergeleitet. Als zweites betrachten wir Protokolle, um die Knoten des Netzwerks zu färben. Wir analysieren einfachstmögliche Färbungsalgorithmen, bei denen jeder Knoten seine Farbe nur aufgrund der Farben oder Bezeichner (Namen) seiner direkten Nachbarn wählt. Als drittes studieren wir die verteilte Komplexität von Problemen auf speziellen Netzwerktopologien, welche im Zusammenhang mit drahtlosen Netzwerken wie Ad Hoc- oder Sensornetzwerken vorkommen.

Acknowledgements

This thesis would not have been possible without the help and support of many people to whom I would like to express my gratitude. First of all, I would like to thank my advisor Roger Wattenhofer for the enourmous effort he put into me, allowing me to be his second Ph.D. student. We had many inspiring research discussions resulting in a great number of ideas which form the basis of this thesis. I count myself extremely lucky that we met in the cafeteria on that Wednesday noon in fall 2001.

I would also like to acknowledge the work of my co-examiners Nati Linial and Friedhelm Meyer auf der Heide. It was a great honor that you agreed to join my thesis committee and I certainly enjoyed your very positive comments.

I am grateful to all professional collegues who enriched my life during the last three and a half years. Particular thanks of course go to the members of the Distributed Computing Group. I would like to thank Aaron for being an exceptionally pleasant and patient office mate during the whole time, for successfully working together on a number of different topics, and for always offering his time to proofread most of my English texts—including this thesis—; Keno for being a good companion during many night shifts and for being a great source for all the software I needed; Regina for sharing the interest in any kind of abstract mathematical theory, for her great sarcastic kind of humor, and also for proofreading many of my English manuscripts—including this thesis as well—; Thomas for being a partner for extremely productive scientific work and especially for significantly contributing to and co-authoring many of my most important publications; Pascal for his good sense of humor and for letting me co-author the MMSC paper—I think I made a real contribution to that paper in the meanwhile—; Nicolas for strengthening the Basle fraction of our group, for bringing along "Ueli" beer several times—don't leave Basle without having tried it—, and for being a great company on many train trips between Basle and Zurich; Stefan for agreeing to look at the dynamic peer-to-peer problem, I wanted to study almost from the beginning of my thesis and for bringing a lot of real power into our tablesoccer game; Andi for helping with many hardware and software problems—especially if I did not read all the TIK emails carefully enough—; the new group members Roland, my brother Michael, and Yves. A special thank goes to Tim Nieberg for the extremely interesting and productive scientific work during his two visits. Of course, I am also grateful to all the students I had the pleasure to advise in their theses. I would like to thank my Master and Diploma students Philipp Boksberger, Thomas Moscibroda, Thomas Rusterholz, Stefan Schmid, and Joest Smit and my "semester students" Andreas Westhoff, Yan Zhang, Thomas Rusterholz, Raphael Boog, and Josias Thoeny. It is fantastic that two of them—Thomas Moscibroda and Stefan Schmid—joined the Distributed Computing Group as Ph.D. students after their theses.

Last but definitely not least, I would like to express my greatest gratitude to my family. This work would not have been possible without my parents Anne-Marie and Herbert. During all the years, you always stood behind me and supported me wherever possible. I am also deeply indebted to my wife Fränzi for her love and encouragement. In busy times—during a Ph.D. thesis, this is almost always the case—, you were an enormous help by taking over many everyday duties. You also know how to put me under pressure when necessary. Without you, this thesis might not be completed by now. Finally, I am grateful to my brother Michael who keeps our family in the Distributed Computing Group and to Fränzi's sons Luc—our system administrator at home—and David who always supported me and who tolerated the sometimes peculiar working hours of a Ph.D. student.

Contents

Chapter 1

Locality in Distributed Computations

1.1 Achieving a Global Goal Based on Local Information

Many of the most fascinating and fundamental systems in the world are large and complex networks, such as the human society, the Internet, or the brain. Such systems have in common that their entirety is composed of a multiplicity of individual *entities*: human beings in society, hosts in the Internet, or neurons in the brain. As diverse as these systems may be, they share the key characteristic that the capability of direct communication of each individual entity is restricted to a small subset of neighboring entities. Most human communication, for instance, is between acquaintances or within the family, and neurons are directly linked with only a relatively small number of other neurons for neuro-transmission. On the other hand, in spite of each node being restricted to *local communication*, the entirety of the system is supposed to come up with some kind of global solution, or achieve a global equilibrium.

Achieving a *global goal* based on *local information* is challenging. Many of the systems which are the focus of computer science fall exactly into the above mentioned category of networks. In the Internet, large-scale peer-to-peer systems, or mobile ad hoc and sensor networks, no node in the network is capable of keeping global information on the network. Instead, these nodes have to perform their intended (global) task based on local information only. In other words, all computation in these systems is *local computation*! It therefore seems natural to systematically study the possibilities and limitations of local computations. What kind of global tasks can be performed by individual entities that have to base their decisions on local information, or how much local information is required in order to come up with a globally optimal solution? Clearly, there are problems—such as computing a spanning tree of a network—which cannot

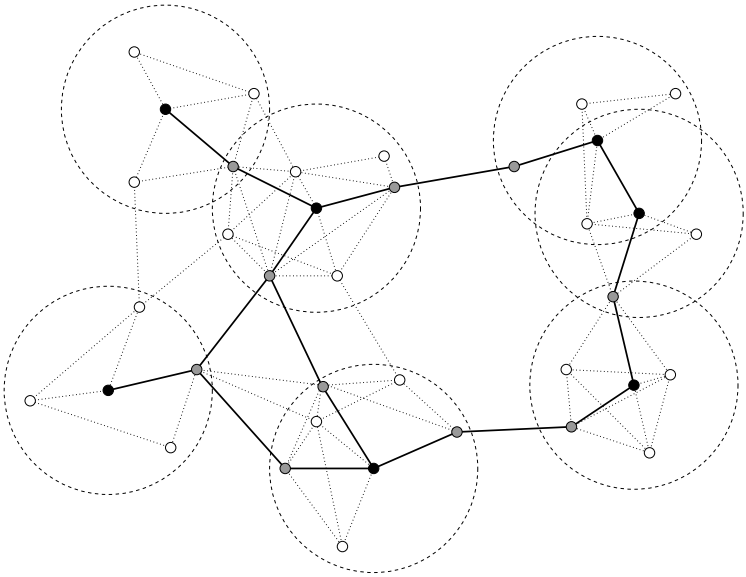

Figure 1.1: Using a dominating set to construct a routing backbone for a given graph G: The black nodes form a dominating set of G (dashed circles illustrate possible assignments of nodes to dominators). The nodes of the dominating set are extended to a connected sub-graph of G using the gray bridge nodes. Black and gray nodes together form a connected dominating set which can be used as routing backbone on G.

be computed locally. It is not even possible to find out locally whether a given set of edges forms a spanning tree of the network graph. However, for many tasks—such as coloring a graph with a given number of colors t—, it is at least possible to check feasibility by only looking at neighboring nodes. Does local information also suffice to assign colors to nodes such that a proper t-coloring of the network graph is obtained? For a given problem, we need to be able to locally check the quality or feasibility of a solution in order to locally compute such a solution. We will see that there is a plethora of important problems which have such a locality condition. Finding out whether it is possible to locally solve a given locally checkable network problem is the main objective of this work. To get some intuition of the nature of the problems considered in this thesis, we look at a few examples arising in the context of large and complex computer networks in the following.

1.1.1 Clustering in Ad Hoc and Sensor Networks

The first example is the problem of clustering a wireless ad hoc or sensor network. Ad hoc and sensor networks consist of wireless devices which communicate via radio without stationary server infrastructure. In sensor networks, small sensor nodes are deployed in some geographic area to measure and collect information of some kind about the respective area. When sending a message from one ad hoc or sensor node to another, intermediate nodes have to serve as routers. Due to the scarcity of energy and the mobility of nodes in ad hoc networks, traditional routing protocols as applied in wired networks fail. In fact, although a number of interesting suggestions have been made (e.g. [23, 70, 84, 114, 120]), finding efficient algorithms for the routing process remains the most urgent problem in the area of ad hoc and sensor networks.

Grouping nodes into clusters [15, 19, 27, 54, 90] is one effective way to save energy [63] and to improve the performance of routing algorithms in general and of the flooding phase comprised by most protocols in particular [29, 77, 82, 122, 125]. The routing is then done between clusters. Clusters typically consist of a cluster-head and a number of adjacent nodes. Naturally, we would like to minimize the number of clusters under the condition that every node of the network belongs to a cluster. Thus, we want to find a smallest possible set of cluster-heads such that every non-cluster-head has a cluster-head in its neighborhood, a problem formally known as the minimum dominating set problem [3, 26, 51, 78, 79, 128, 133]. To enable routing between clusters, cluster-heads usually connect to each other by selecting additional bridge nodes (see Figure 1.1). Alternatively, we can choose cluster-heads such that the topology induced by them is connected. The routing backbone which is obtained in both cases is a structure which is formally known as a connected dominating set. How close can we get to an optimal clustering if nodes are only allowed to communicate (directly or indirectly) with nodes up to distance k? Or, what we will see to be equivalent: what is the optimal trade-off between the quality of the solution and the running time of an algorithm?

1.1.2 More Distributed Optimization Problems

The second closely related example originates from the area of wireless sensor networks. Each node of such a network has a particular region which it can sense for the desired information. Usually, those sensing regions overlap, prolonging the lifetime and improving the reliability of the network. However, we can only get the network to live longer if redundant nodes can be turned off most of the time. It is therefore desirable to find a small subset of all sensors which cover the whole area [48] or, even better, to find subsets of sensors such that the lifetime of the network is maximized [103].

More generally, we could think of the following abstract scenario. Assume that in a large network such as the Internet or a wireless network we have a set S of nodes which are able to provide a particular service to the network. Each

node of S might be able to provide the given service to at most γ nodes in the network which are at distance at most r. We have to choose a subset S' of S and an assignment of nodes from S' to network nodes such that all nodes of the network are served by at least t nodes. Thereby, the goal is to minimize the size of S', the maximum number of nodes of S' in the r-neighborhood of a network node, or to optimize any other reasonable quality measure. Given such a global optimization problem with local feasibility conditions, how good can a solution be which is obtained by a local, distributed algorithm?

1.1.3 Network Labelings

The last examples consider a different type of problem. For all the problems described so far, finding a feasible solution is easy but it is challenging to locally find a solution which is a good approximation of an optimal solution. For many important problems, it is already hard to obtain feasible solutions. In a class of problems which we call labeling problems, we want to label nodes with a given number of labels such that a set of local constraints is satisfied [105]. Computing a maximal independent set (MIS), that is, finding a dominating set of non-adjacent nodes of the network is the most prominent example of this kind. While sequentially, an MIS can be found by a simple greedy algorithm (add independent nodes as long as possible), efficiently constructing an MIS by a distributed algorithm turns out to be a non-trivial task. In fact, distributed MIS computation can be seen as a Drosophila of distributed computing as it prototypically models symmetry breaking [72], the problem of assigning different roles to nodes which have symmetrical views of the network [9, 33, 91, 96]. Besides being theoretically interesting, due to the special topologies of ad hoc and sensor networks, maximal independent sets are good clusterings of such networks [3, 26, 98, 128]. Especially if the clustering is used as a basis for a TDMA or FDMA scheme[1], it is beneficial if cluster-heads are non-adjacent [18, 19, 99, 102]. When constructing a TDMA or FDMA scheme, the ultimate goal is to find a coloring of the network. Similar to the MIS problem, graph coloring has been used as a prototypical model for symmetry breaking. The distributed complexity of graph coloring is therefore also considered fundamental for the general understanding of distributed systems, a fact which resulted in a multiplicity of papers considering coloring in different distributed models [9, 33, 55, 91, 100].

[1]TDMA stands for time division multiple access whereas FDMA stands for frequency division multiple access. They both are means to share a common communication medium between different network devices. In an FDMA scheme, every network node gets assigned a frequency such that nodes which interfere when sending simultaneously have different frequencies. In a TDMA scheme, time is divided into slots such that possibly interfering nodes use different time slots.

1.2 Computational Model

Throughout this thesis, we use the standard message passing model. The network is modeled as an undirected and in most cases unweighted graph $G = (V, E)$. Two nodes $u, v \in V$ of the network are connected by an edge $(u, v) \in E$ whenever there is a direct bidirectional communication channel connecting u and v. We usually assume that each node u has a unique identifier $\mathrm{ID}(u)$ of size $\mathrm{O}(\log n)$ where $n := |V|$ is the number of nodes of G.

For simplicity, we assume communication to be synchronous throughout the thesis. Typically, in synchronous systems all nodes wake up (or start an algorithm) simultaneously. Time is divided into rounds. In each round, every node can send a message to each of its neighbors and perform some local computation based on the information contained in the messages of the previous rounds. The *time complexity* of a synchronous distributed algorithm is the number of rounds until all nodes terminate. The *message complexity* of a distributed algorithm is defined as the total number of messages sent by the algorithm. Since in distributed systems the running time of an algorithm is mainly determined by the time needed for the communication, we make no restrictions on local computations. In principle, in each round our model allows each node to compute an arbitrary (computable) function. However in all proposed algorithms, unless explicitly stated, all local computations are reasonably small. Note that the synchronous wake-up condition is merely a simplification and does not restrict the computational power of the model. If we drop this condition but assume that receiving a message wakes up a node, we obtain a model which is equivalent up to constant factors. In this case, time complexity must be defined as the maximal time from a node's wake-up until its termination. Further, instead of dividing time into rounds, it is equivalent to assume that the time for sending a message over an edge of G is bounded by some globally known constant T. After waiting long enough (T), a node u can be sure that all messages of the current round are received, that is, u can proceed to the next round.

Working with the above described synchronous communication model, two fundamental *obstacles* arise in the design of distributed algorithms, namely, *locality* and *congestion*. Similar to [112], we distinguish two prototypical models, $\mathcal{LOCAL}$ and $\mathcal{CONGEST}$, depending on how much information can be sent in each message. The two models allow us to understand the adverse effects of locality and congestion on time and message complexities of distributed algorithms.

The $\mathcal{LOCAL}$ model provides a tool to analyze the effects of locality on distributed computations. It abstracts away all other restricting factors. In particular, congestion has no influence on time or message complexity. In the $\mathcal{LOCAL}$ model (e.g. [91, 105, 112]), knowing one's k-neighborhood and performing k communication rounds are equivalent. It is assumed that in every communication round each node in the network can send an *arbitrarily long message* to each of its neighbors. Because messages are unbounded, in k communication rounds a node v may collect the IDs and interconnections of all

nodes up to distance k from v. Hence, each node has a partial (local) view of the graph; it knows its entire vicinity up to distance k. Let $\mathcal{T}_{v,k}$ be the topology seen by v after k rounds. $\mathcal{T}_{v,k}$ is the graph induced by the k-neighborhood of v without all edges between nodes at distance exactly k. The labeling (i.e., the assignment of IDs) 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 can do in time k is to have every node v collect its k-neighborhood and base its decision on $\mathcal{V}_{v,k}$. Since the $\mathcal{LOCAL}$ model focuses entirely on the localized aspects of distributed computations and abstracts away all other aspects arising in the design of distributed algorithms, it is the most fundamental model when studying the phenomenon of locality, particularly for lower bounds.

The goal of the $\mathcal{CONGEST}$ model is to understand the limiting effect of the amount of necessary communication on the time and message complexities of distributed algorithms. In practice, the amount of information exchanged between two neighbors in one communication step is limited. The $\mathcal{CONGEST}$ model [50, 112] takes into account the *volume of communication*. This model limits the information that can be sent in one message to $\mathrm{O}(\log n)$ bits. Apart from constraining the maximal message size, the $\mathcal{CONGEST}$ model is equal to the $\mathcal{LOCAL}$ model. Given this additional restriction, even problems on the complete network graph, which could be solved optimally in a single communication round in the $\mathcal{LOCAL}$ model, become nontrivial [95].

In the described $\mathcal{LOCAL}$ and $\mathcal{CONGEST}$ models, we assume that in each round every node can send a distinct message to each of its neighbors. In practice, this is not always realistic. Particularly in wireless networks, a common model assumes that nodes sending a message actually broadcast the message to all immediate neighbors. We therefore define models $\mathcal{LOCAL}_{\mathrm{BC}}$ and $\mathcal{CONGEST}_{\mathrm{BC}}$ which differ from the $\mathcal{LOCAL}$ and $\mathcal{CONGEST}$ models only in the way that in each round every node has to send the same message to all its neighbors. It is not hard to see that only allowing to locally broadcast a message is no restriction in the case of the $\mathcal{LOCAL}$ model, that is, the computational powers of $\mathcal{LOCAL}$ and $\mathcal{LOCAL}_{\mathrm{BC}}$ are the same. In fact, instead of sending distinct messages to all its neighbors, a node can pack all the messages into one large message which is then broadcasted to the neighbors. It would even be conceivable to further restrict the communication in each round for example by only allowing each node to send a message to exactly one of its neighbors. However, in the context of this thesis we stick to the described models.

To end this section, we would like to shortly discuss two other fundamental issues which are not part of the described models, namely *asynchrony* and *randomization*. We have deliberately abstracted away all problems arising in asynchronous systems in our communication models. Besides simplicity the major reason is that we mainly focus on time complexity rather than message complexity throughout the thesis. The common definition of asynchronous communication assumes that each sent message arrives after a finite but completely unpredictable time. The time complexity of an algorithm is the time of a worst-case execution where for the analysis a maximum message delay of 1 time unit

is assumed. If the number of sent messages is no issue, there is a simple method to locally synchronize an asynchronous system. In every 'round', every node sends a message to all its neighbors, no matter whether all those messages are needed in the synchronous protocol. As soon as a node u has received the messages of a given round from all of its neighbors, u can proceed to the next round. In Awerbuch's seminal work on network synchronization [5], the described method is called an α-synchronizer. In [5, 10], it is described how, at the cost of a poly-logarithmic factor in time complexity, also the message cost of network synchronization can be kept within a poly-logarithmic factor of the message complexity of the synchronous algorithm.

Understanding the role of randomization in distributed message passing systems is extremely fundamental. The most important usage of randomization in distributed algorithms arises in the context of symmetry breaking. On the one hand, breaking symmetries using randomized solutions is often possible with small overhead [2, 68, 93, 96]. On the other hand, deterministic symmetry breaking is notoriously hard and the obtained solutions are far less efficient and considerably more complicated [9, 60, 61, 108]. Even though we have not specified the allowed amount of randomization in our models, randomization is an important issue of this work. Throughout the thesis, we explicitly quantify and analyze the effects of randomization in many interesting cases.

In the following two sections, we summarize fundamental work for the described communication models. Therein we will have a closer look at the main difficulties arising in the design of distributed algorithms in the $\mathcal{LOCAL}$ and $\mathcal{CONGEST}$ models, respectively.

1.3 The $\mathcal{LOCAL}$ Model: The Power of Knowing the k-Neighborhood

The $\mathcal{LOCAL}$ model in the given form (synchronous model with unbounded messages and local computations) has been introduced by Linial in the first paper which is devoted to the fundamentals of local computations [91]. In 2005, many of the results of [91] are still state-of-the-art almost twenty years after their first publication in 1987. The essential question concerning the $\mathcal{LOCAL}$ model has been raised by Naor and Stockmeyer in the title of their paper 'What Can Be Computed Locally?' [105]. In [105], it is shown that for certain graphs, there are locally checkable labelings which can be computed in a constant number of rounds. When developing algorithms for the $\mathcal{LOCAL}$ model, we face two main problems.

1. *Locality of the distributed problem*: How good can a global solution be which is composed of optimal local solutions?

2. *Symmetry breaking*: How can all nodes decide on a common decomposition into local problems such that different local solutions do not interfere?

Most of the problems which were introduced in Section 1.1 can be decomposed into local problems such that the combined global solution is reasonably close to an optimal solution. Two exemplary problems which cannot be solved locally are the minimum spanning tree (MST) problem and the minimum capacitated vertex cover problem [57, 112]. In a capacitated vertex cover, each edge is assigned to one of its two nodes such that no node gets assigned more edges than a given value called the node's capacity. For both problems, the ring network is a bad example. An MST of a ring consists of all edges of the ring but the largest one. If we want to compute a capacitated vertex cover with all node capacities set to 1, all nodes of a ring have to decide on a common direction in order to be able to cover all edges. However, finding the largest edge of a ring or finding a common direction is not possible without seeing the whole ring. Similarly, one can also show that a ring with an even number of nodes cannot be 2-colored without seeing the whole ring [91].

1.3.1 Maximal Independent Sets and Colorings: Modeling Symmetry Breaking in Distributed Systems

While decomposability into local problems is rather problem-specific, symmetry breaking can to some extent be considered independently of a particular problem. Two problems which have a special role in the context of symmetry breaking are the construction of a maximal independent set (MIS) and the computation of a $(\Delta+1)$-coloring. An independent set is a set of pairwise non-adjacent nodes of a graph. An MIS is an independent set S such that for every node u which is not in the MIS, there is an adjacent node $v \in S$. A $(\Delta+1)$-coloring is an assignment of colors $1,\ldots,\Delta+1$ to the nodes of a graph such that no two neighbors have the same color. Both structures can be computed by simple sequential algorithms. In the distributed setting, both problems prototypically model the main difficulties arising when symmetries have to be broken. We therefore start our discussion of previous work in the $\mathcal{LOCAL}$ model with these two problems.

The best distributed MIS algorithm is a simple randomized algorithm with expected time complexity $O(\log n)$ [2, 96]. In [2, 96], the algorithm has been described for the PRAM model and not for the distributed computing model described in Section 1.2. Although on a PRAM, there are no locality issues, the algorithm can directly be adapted to the $\mathcal{LOCAL}$ and even to the $\mathcal{CONGEST}$ and $\mathcal{CONGEST}_{\mathrm{BC}}$ models. In [91], a nice reduction from $(\Delta+1)$-coloring to the MIS problem is described. For a given graph G which we want to color with $\Delta+1$ colors, a graph G' is constructed as follows. We make $\Delta+1$ copies $v_0,\ldots,v_\Delta$ for every node v of G. All $\Delta+1$ copies are connected to form a clique. Two nodes u_i and v_j of G' are connected if $(u,v) \in E(G)$ and if $i=j$. Then an MIS on G' is computed by a given distributed algorithm. For every node v of G, exactly one of the $\Delta+1$ copies $v_0,\ldots,v_\Delta$ is in an MIS of G'. If it is v_i, we assign color i to node v. Because $(u,v) \in E(G)$ implies that u_i and v_i cannot both be in the MIS, this gives a $(\Delta+1)$-coloring of the original graph.

Applying this reduction with the algorithm of [2, 96] results in a randomized $(\Delta + 1)$-coloring algorithm with expected time complexity $O(\log n)$.

In the previous section, we have pointed out that randomization plays a fundamental role in symmetry breaking. It is therefore no surprise that finding an MIS or a $(\Delta + 1)$-coloring with a deterministic algorithm turns out to be a lot harder. There are special graphs for which extremely efficient deterministic algorithms exist. On a ring or on a rooted tree, it is possible to compute an MIS or a 3-coloring in time $O(\log^* n)$ [33, 55]. Using the algorithm of [33, 55], it is also possible to $(\Delta+1)$-color a constant-degree graph in time $O(\log^* n)$. Those upper bounds are matched by a lower bound in [91], stating that $\Omega(\log^* n)$ rounds are needed to compute an MIS or a coloring with a constant number of colors for the ring. Note that the lower bound also holds for randomized algorithms. For general graphs, the best deterministic MIS and $(\Delta + 1)$-coloring algorithms are based on a structure called network decomposition. Network decompositions can be used as a general tool for breaking symmetries for a great number of different local problems which go beyond MIS and coloring. We will describe this approach in the following section.

1.3.2 Network Decomposition: A General Tool for Distributed Algorithms

We have introduced symmetry breaking as the problem of choosing a common decomposition of a global problem into local problems such that we can compute a global solution by combining optimal solutions of all local problems. Network decompositions provide a problem-independent method for decomposing a distributed problem. The idea is to decompose the network graph into clusters of small diameter. These clusters then define the local problems which have to be solved.

Let $G = (V, E)$ be the network graph. A cluster $C \subseteq V$ is a subset of the nodes of G such that the sub-graph induced by the nodes in C is a connected component of G. The *strong diameter* of a cluster C is defined as the diameter of the sub-graph of G induced by the nodes in C. Let $d_G(u, v)$ be the length of a shortest path connecting u and v on G. The *weak diameter* of a cluster C is the maximum distance $d_G(u, v)$ in the original graph G for any two $u, v \in C$. Note that the weak diameter of a cluster C is upper-bounded by the strong diameter of C. A network decomposition of G is now defined as follows.

Definition 1.1. (Network Decomposition) *A (χ, d)-decomposition of a graph G is a partition of G into disjoint clusters and a χ-coloring of the clusters such that*

1. *The strong diameter of each cluster is at most d.*
2. *Two clusters C_1 and C_2 have different colors if they are connected by an edge $(v_1, v_2) \in E$ where $v_1 \in C_1$ and $v_2 \in C_2$.*

A *weak network decomposition* is defined analogously with the only difference that only the weak diameter of each cluster has to be bounded by d. Given a (χ, d)-decomposition, an MIS or $(\Delta+1)$-coloring can be computed as follows. The protocol works in χ phases. In phase i, all clusters of color i compute their part of the global solution. In the case of the MIS, a cluster adds as many of its nodes as possible to the global MIS. In the case of $(\Delta+1)$-coloring, a cluster assigns one of the $\Delta+1$ colors to each of its nodes such that there is no conflict with colors assigned by other clusters. Because each cluster has diameter at most d and because clusters of the same color do not interfere, each phase can be computed in $\mathrm{O}(d)$ time. Each cluster can for example select a leader which collects all the data of the nodes of the cluster and of adjacent nodes of neighboring clusters. The leader can then locally (without communication) compute a solution of its cluster and afterwards broadcast this solution to all nodes of the cluster. Given a (χ, d)-decomposition, it is therefore possible to compute an MIS or a $(\Delta+1)$-coloring in time $\mathrm{O}(\chi \cdot d)$. In a similar way, network decompositions can be used to solve a large number of different distributed problems. In $\mathrm{O}(d)$ time, one can for instance compute χ-approximations for minimum graph coloring or minimum dominating set. In this case, each cluster computes an optimal local coloring or dominating set in $\mathrm{O}(d)$ time. Note that we explicitly allowed exponential local computations when defining the $\mathcal{LOCAL}$ model. The combination of the local solutions of all clusters of the same color are at least as good as a global optimal solution because those clusters do not interfere. Because there are only χ colors, combining the local solutions of all clusters results in a χ-approximation.

The given notion of a network decomposition was first introduced in [9] where it is shown that for $\alpha \in \mathrm{O}(\sqrt{\log\log n / \log n})$ it is possible to deterministically compute an (n^α, n^α)-decomposition in time n^α. This result has been improved in [108] where a deterministic algorithm to compute a $(n^{\mathrm{O}(\beta)}, n^\beta)$-decomposition in time $n^{\mathrm{O}(\beta)}$ for $\beta = \sqrt{1/\log n}$ is described. Using the decomposition algorithm of [108], we obtain the fastest known deterministic algorithms for MIS construction and $(\Delta+1)$-coloring. Note that the time complexity $n^{\mathrm{O}(\sqrt{1/\log n})}$ is much faster than linear but still far away from the $\mathrm{O}(\log n)$-time complexity of the randomized algorithm of [2, 96]. Whether there is a distributed deterministic MIS algorithm with poly-logarithmic running time remains an important and demanding open problem [92].

In [93], it is proven that for $p \in (0,1)$ and for $\chi = \log n / \log(1/(1-p))$, there is a (χ, d)-decomposition with $d \leq 2\log n / \log(1/p)$ for every graph G. This implies that for $d \in \mathrm{O}(\log n)$, there is a (χ, d)-decomposition with $\chi \leq n^{1/d} \log n / 2$ and that for $\chi \in \mathrm{O}(\log n)$, there is a (χ, d)-decomposition with $d \leq 2n^{1/\chi} \log n$. In particular for every graph G, there is a $(\log n, 2\log n)$-decomposition. It is also shown in [93] that there are graphs for which the above results are essentially tight.

In addition to the above existence results, [93] also contains a randomized distributed algorithm which computes a weak network decomposition that es-

sentially matches the given bounds. The algorithm consists of χ phases which each need $O(d)$ rounds. In phase i, the clusters of color i are constructed. The time complexity of the distributed weak decomposition algorithm therefore is $O(\chi \cdot d)$. In the $\mathcal{LOCAL}$ model, this time complexity can be reduced to $O(d)$ because in principle all colors can be computed simultaneously.

We have introduced network decompositions as a tool to build distributed algorithms. In [8], network decompositions have been used to obtain better network decompositions. It is shown how a weak-diameter decomposition can be turned into a strong-diameter decomposition and how to compute $(kn^{1/k}, 2k)$-decompositions using the techniques of [9, 108]. In particular, by combining [8] and [93], an $(O(\log n), O(\log n))$-decomposition can be computed in poly-logarithmic time by a randomized algorithm. Combining [8] and [108] yields an $O(n^{O(\sqrt{1/\log n})})$-time deterministic algorithm for the same result.

Network decompositions have also been considered in other parallel computing models. On a PRAM, a $(\log^2 n, \log n)$-decomposition can be computed in poly-logarithmic time [7]. Furthermore, in the literature on distributed computing different decompositions of graphs into regions of small diameter have been used in various problems such as network synchronization [5, 10], routing [11, 12], spanner constructions [112], or tracking of mobile users [11, 13].

1.3.3 Problem-Specific Distributed Algorithms

Apart from the work about network decompositions which systematically analyzes the properties of the $\mathcal{LOCAL}$ model, a variety of distributed algorithms for different specific problems have been published as well. In the following, we summarize the most important of those results.

Closely related to the MIS problem is the problem of constructing a maximal matching (MM), which is a maximal set of pairwise non-adjacent edges. Analogously to the MIS case, an MM can be computed by a randomized algorithm in time $O(\log n)$ [2, 68, 96]. Surprisingly, it is also possible to deterministically compute an MM in poly-logarithmic time [60, 61]. All MM algorithms are 2-approximations for the maximum matching problem. A randomized algorithm which 5-approximates the weighted maximum matching problem in time $O(\log^2 n)$ has been presented in [130].

We have discussed distributed algorithms to compute $(\Delta + 1)$-colorings of the network graph in Sections 1.3.1 and 1.3.2. Let us now look at more general graph coloring problems. In [91], a deterministic algorithm which colors any graph with $O(\Delta^2)$ colors in $O(\log^* n)$ rounds is presented. Because any t-coloring can be turned into a t' coloring in $t - t'$ rounds, the algorithm implies an $O(\Delta^2 + \log^* n)$-time algorithm for computing a $(\Delta + 1)$-coloring, a result which has also been presented in [55]. Also interesting for graphs with moderate degrees is an algorithm which finds a $(\Delta + 1)$-coloring of an arbitrary graph in time $O(\Delta \log n)$ [9, 55, 112]. The techniques which are applied to achieve this can even be generalized to transform any given t-coloring into a $(\Delta + 1)$-coloring in time $O(\Delta \log t)$. Combined with the $O(\Delta^2)$-coloring algorithm of

[91], this gives an $O(\Delta \log \Delta + \log^* n)$-time $(\Delta+1)$-coloring algorithm for general graphs. In fact, the same method can also be applied to construct an MIS in time $O(\Delta \log \Delta + \log^* n)$. New interesting ideas for the problem of coloring a graph by a local algorithm have been presented in [35]. In [35], it is claimed that the presented algorithm colors any graph with $O(\Delta)$ colors in $O(\log^* n)$ rounds. Unfortunately, there is an error in the analysis of [111]. With a correct analysis, we can only show that the given algorithm achieves an $O(\Delta^2)$-coloring in $O(\log^* n)$ time, that is, asymptotically, the algorithm is not better than the one of [91].

Similarly to the MIS versus MM case, there are stronger results for edge colorings than for the more general vertex colorings. Because an edge coloring of a graph G corresponds to a vertex coloring on the so-called line graph of G, all vertex coloring algorithm can also be applied for to color edges. The maximum degree of the line graph can be as much as $2\Delta - 2$. Applying the randomized $O(\log n)$-time algorithm for $(\Delta + 1)$-colorings to the edge coloring problem, we therefore obtain an $O(\log n)$ time algorithm which colors the edges of a given graph with $2\Delta - 1$ colors. In [109], a randomized algorithm which colors the edges of a graph with $1.6\Delta + O(n^{1+\varepsilon})$ colors in poly-logarithmic time has been presented. A further improvement has been given in [56]. For any constant $\varepsilon > 0$, a $(1 + \varepsilon)\Delta$-edge coloring of a graph G can be computed by a randomized algorithm in time $O((1 + 1/c) \log \log n)$ if the minimum degree of G is $\Omega(n^{c/\log\log n})$. There is also a deterministic edge-coloring algorithm which colors the edges of a graph with $O(\Delta \log n)$ colors in poly-logarithmic time [34].

The first important 'local' problem described in Section 1.1 is clustering of ad hoc and sensor networks. We have seen that computing a clustering essentially boils down to finding a small (connected) dominating set. The best distributed algorithms for the minimum dominating set problem are described in [69] and [118]. The algorithm of [118] has been formulated for the PRAM model. It is, however, possible to use the same techniques in a distributed model. In [69], a randomized algorithm which in expectation computes an $O(\log \Delta)$-approximation in time $O(\log \Delta \log n)$ with high probability is presented . Similar results are achieved for the connected dominating set problem in [36]. For a more detailed discussion of distributed algorithms related to dominating sets, we refer to Chapters 2 and 3.

1.4 The $\mathcal{CONGEST}$ Model: The Role of Bandwidth Restrictions

In order to analyze the restrictions incurred by the bounded message size in the $\mathcal{CONGEST}$ model, we would like to have a problem, analogous to MIS and coloring for symmetry breaking, which prototypically models congestion problems. We therefore need a problem for which locality is not the restricting factor. It turns out that the MST problem is well-suited for this purpose although we have seen that computing an MST is inherently not local. In the

$\mathcal{LOCAL}$ model, computing an MST needs time $\Theta(D)$ where D is the diameter of the network graph. The first distributed MST algorithm for the $\mathcal{CONGEST}$ model is by Gallager, Humblet, and Spira [50]. The time complexity of the algorithm of [50] is $O(n \log n)$. This time complexity has gradually been reduced to $O(n \log^* n)$ [30, 49] and $O(n)$ [6] in the following. The first sub-linear time algorithm has been presented in [52]. It has time complexity $O(D+n^{0.613} \log^* n)$. This has later been improved to $O(D + \sqrt{n} \log^* n)$ in [86]. The last algorithm has been proven to be almost optimal by Peleg and Rubinovich who showed that for $D \in \Omega(\log n)$, every distributed MST algorithm has time complexity at least $\Omega(D + \sqrt{n}/\log^2 n)$ in the $\mathcal{CONGEST}$ model. This lower bound has been generalized and improved in [39] where it is shown that computing an α-approximation for the MST needs time at least $\Omega(D + \sqrt{n/(\alpha \log n)})$. For graphs of constant diameter, the MST problem is studied in [94] and [95]. On the one hand, it is shown in [94] that the time required to compute an MST for graphs of diameter 4 and 3 is $\Omega(\sqrt[3]{n}/\log n)$ and $\Omega(\sqrt[4]{n}/\sqrt{\log n})$, respectively. On the other hand, [94] gives an $O(\log n)$ time algorithm for diameter 2 graphs. For diameter 1 graphs, that is, for the weighted complete graph, there is an algorithm with time complexity $O(\log \log n)$ [95]. Note that if the diameter is constant, an MST can be computed in time $O(1)$ in the $\mathcal{LOCAL}$ model.

Apart from the MST problem, there is not a lot of systematic work about the fundamental limitations of the $\mathcal{CONGEST}$ model. However, many of the distributed algorithm for the problems discussed in Section 1.3 can actually be applied in the $\mathcal{CONGEST}$ or even in the $\mathcal{CONGEST}_{\mathrm{BC}}$ model. Examples are the randomized algorithms for independent sets and matchings [2, 68, 96, 130], the coloring algorithm of [33, 55, 91], or the minimum dominating set algorithm of [69]. In contrast to the $\mathcal{LOCAL}$ model, all these algorithms remain the best known algorithms even if the diameter of the network graph is constant. Analyzing the complexity of these problems for small-diameter graphs in the $\mathcal{CONGEST}$ model is an interesting open problem.

1.5 Definitions and Preliminaries

As described in Section 1.2, we model networks by undirected graphs. As far as possible, we try to use consistent notations for describing graphs and their properties throughout the thesis. Unless explicitly stated, the network graph is called G and its node and edge sets are called $V(G) = G$ and $E(G) = E$, respectively. Further, n denotes the number of nodes, Δ the maximum degree, and D the diameter of G, respectively. The distance measured in hops between two nodes u and v is denoted by $d_G(u, v)$. As we deal with local algorithms, we will often use neighborhoods of a particular depth. We use the following definitions:

$$\Gamma_r(v) \;:=\; \big\{u \in V \,\big|\, d_G(u,v) = r\big\} \quad \text{and} \quad \Gamma_r^+(v) \;:=\; \bigcup_{i=0}^{r} \Gamma_i(v).$$

Therefore, $\Gamma_r(v)$ is the set of nodes at distance exactly r from v and $\Gamma_r^+(v)$ is the set of nodes at distance at most r from v. For convenience, we also define

$$\Gamma(v) := \Gamma_1(v) \quad \text{and} \quad \Gamma^+(v) := \Gamma_1^+(v) = \{v\} \cup \Gamma(v).$$

The degree of a node v is denoted by $\delta(v) := |\Gamma(v)|$. Arguing about local algorithms is considerably simplified if the local views have a structure which is not arbitrarily complex. Especially in the context of lower bounds, graphs which locally look like trees will prove extremely useful. In order to have trees in each k-neighborhood of a graph G, G must not have cycles of length at most $2k$. The shortest cycle of a graph G is called the girth $g(G)$ of G. The following well-known theorem shows an important relation between the number of edges $|E|$ and the girth $g(G)$ of a graph $G = (V, E)$ (see e.g. [22]).

Theorem 1.1. *Let $G = (V, E)$ be a graph with girth $g(G) \geq k$. There is a constant c such that he number of edges of G is at most $|E| \leq n^{1+c/k}$. Further, there are n-node graphs with more than $n^{1+d/k}$ edges and girth at least k for some constant $d > 0$.*

For convenience, we define the set of all integers between 1 and k as

$$[k] := \{1, \ldots, k\} := \big\{i \in \mathbb{Z} | 1 \leq i \leq k\big\}.$$

Depending on the context, we will use the notations $[k]$ and $\{1, \ldots, k\}$ interchangeably to denote the given set.

When discussing MIS and coloring algorithms in Section 1.3, we have already used the so-called log-star function $\log^* n$. Let $\log^{(i)} n$ be the value which results after applying the logarithm i times to n. The function function $\log^* n$ is defined as

$$\log^* n := \min_i \big(\log^{(i)} \leq 1\big).$$

If not explicitly specified, we always consider logarithms to base 2. The natural logarithm of a number x is denoted by $\ln x$.

Throughout the thesis, we will consider distributed optimization problems and algorithms which compute solutions to those problems. To measure the quality of an achieved solution, it is compared to an optimal solution. Assume that we have an algorithm solving some given minimization problem. Let $\mathcal{ALG}_\mathcal{I}$ be the value of the solution produced by the given algorithm for input $\mathcal{I}$. The value of an optimal solution for input $\mathcal{I}$ is denoted by $\mathcal{OPT}_\mathcal{I}$. The approximation ratio ρ of the given algorithm is defined as

$$\rho := \max_\mathcal{I} \frac{\mathcal{ALG}_\mathcal{I}}{\mathcal{OPT}_\mathcal{I}}.$$

Because we want $\rho \geq 1$, for maximization problems, the approximation ratio is defined as $\rho := \max_\mathcal{I} \mathcal{OPT}_\mathcal{I}/\mathcal{ALG}_\mathcal{I}$. If adapting constants in the algorithm allows to achieve an approximation ratio of $1 + \varepsilon$ for every constant $\varepsilon > 0$, the algorithm is called an approximation scheme.

When arguing about randomized algorithms, we will often state that something holds with high probability. In the context of a network graph with n nodes, this means that the probability is at least $1 - 1/n$. Whenever we use the term 'with high probability', it is even possible to reduce the failure probability from $1/n$ to $1/n^c$ for any constant c by adjusting other constants in the algorithm or in the analysis. To bound probabilities, we will frequently use Chernoff bounds. They describe the tail behavior of the distribution of the sum of independent Bernoulli experiments.

Theorem 1.2. (Lower Tail) *Let $X_1, X_2, \ldots, X_N$ be independent Bernoulli variables with $\Pr[X_i = 1] = p_i$. Let $X := \sum_i X_i$ denote the sum of the X_i and let $\mu := \mathrm{E}[X] := \sum_i p_i$ be the expected value for X. For $\delta \in (0, 1]$,*

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

Theorem 1.3. (Upper Tail) *Let $X_1, X_2, \ldots, X_N$ be independent Bernoulli variables with $\Pr[X_i = 1] = p_i$. Let $X := \sum_i X_i$ denote the sum of the X_i and let $\mu := \mathrm{E}[X] := \sum_i p_i$ be the expected value for X. For $\delta > 0$,*

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

Finally, Facts 1.4–1.6 state three important inequalities which we will need at various places.

Fact 1.4. (Means Inequality) *Let $\mathcal{A} \subset \mathbb{R}^+$ be a set of positive real numbers. The product of the values in $\mathcal{A}$ can be upper bounded by replacing each factor with the arithmetic mean of the elements of $\mathcal{A}$:*

$$\prod_{x \in \mathcal{A}} x \leq \left(\frac{\sum_{x \in \mathcal{A}} x}{|\mathcal{A}|}\right)^{|\mathcal{A}|}.$$

Fact 1.5. *For all n, t, such that $n \geq 1$ and $|t| \leq n$,*

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

Fact 1.6. *For integers $n \geq k \geq 1$, we have*

$$\left(\frac{n}{k}\right)^k \leq \binom{n}{k} \leq \left(\frac{en}{k}\right)^k.$$

Chapter 2

Distributed Approximation of Covering and Packing Problems

Many of the introductory example problems of Section 1.1 are classic covering problems or closely related to covering and the dual packing problems. In this and the next chapter, we will study the distributed complexity of covering and packing problems in the communication models described in Section 1.2. Our main objective is a thorough characterization of the possible trade-offs between time complexity (locality) k and the achievable quality of the solution (approximation ratio). We will derive upper bounds (Chapter 2) and lower bounds (Chapter 3) on this trade-off.

Chapter 2 is structured as follows. In Section 2.1, we introduce our notion of distributed covering and packing problems. In particular, we motivate the use of linear programming (LP) relaxation in the context of distributed approximation algorithms. When considering computer networks, the most important covering problem is the minimum dominating set (MDS) problem. Section 2.2 shows how the greedy algorithm for the MDS problem can be turned into a distributed approximation algorithm for the fractional variant of the MDS problem (LP relaxation). The algorithm of Section 2.2 is improved and generalized in Section 2.3 to obtain the main result of this chapter. We present a distributed approximation scheme for all covering and packing LPs for the $\mathcal{CONGEST}_{\mathrm{BC}}$ model.

We have seen that in the $\mathcal{LOCAL}$ model, network decompositions can be applied to obtain fast algorithms for a large class of problems. Based on the randomized weak-diameter network decomposition algorithm of Linial and Saks [93], we show how to compute constant approximations for covering and packing LPs in Section 2.4.

When solving an LP instead of the original combinatorial problem, there needs to be some way to transform a fractional solution into an integer one. For distributed covering and packing problems, this can be achieved by randomized rounding as described in Section 2.5. Finally, in Sections 2.6 and 2.7, we look at the construction of connected dominating sets and discuss randomization issues in the context of distributed covering and packing problems.

2.1 Distributed Covering and Packing Problems

2.1.1 The Minimum Dominating Set Problem

A dominating set of a graph $G = (V, E)$ is a subset $S \subseteq V$ of the nodes of G such that for every node $u \in V$ which is not in S, there is an adjacent node $v \in S$. In Section 1.1, we have seen that dominating sets and the like are important structures which naturally occur in many real-world networking problems. Usually, we aim at having a dominating set which is as small as possible, resulting in the well-known minimum dominating set problem. As we have seen, solving MDS for large-scale networks calls for fast and hence local, distributed algorithms for the problem. In this chapter, we present distributed algorithms which can be applied to many common variants of MDS, the more general covering problems, as well as the closely related packing problems. The main principles behind all described algorithms for general covering and packing problems can be understood by looking only at the MDS problem as a special case. While developing our algorithms, we therefore always first look at the intuitive MDS case before generalizing an idea.

2.1.2 Breaking Symmetries by LP Relaxation

We have argued in the introduction that one of the main challenges arising in the design of distributed algorithms is breaking symmetries. In the case of the MDS problem, on the one hand, we need to guarantee that every node is covered by a node of the dominating set S. On the other hand, in the case of a highly symmetric graph, only some nodes of a set of equally qualified nodes can join the dominating set. Otherwise it is not possible to achieve a good approximation ratio.

The symmetry breaking problem seems to be most involved if all nodes have the same view or similar views. Therefore, the class of regular graphs appears to contain graphs where breaking symmetries is most difficult. However, for regular graphs, there is the following simple distributed MDS algorithm. Assume that the network graph G is δ-regular, that is, all nodes of G have degree δ.

1. Initially, the dominating set S is empty.
2. Each node joins S independently with probability $\ln(\delta + 1)/(\delta + 1)$.
3. If a node u is not covered, it joins S.

If $\mathcal{OPT}$ denotes the size of an optimal dominating set, the described algorithm computes a dominating set S of expected size, that is,

$$\mathrm{E}\left[|S|\right] \;\leq\; (\ln(\delta+1)+1)\,\mathcal{OPT}. \tag{2.1}$$

To show Inequality (2.1), we need the following key observation. Because each node of a dominating set can cover at most $\delta+1$ nodes, we have $\mathcal{OPT} \geq |V|/(\delta+1)$. Hence, the expected number of nodes joining S in step 2 of the above algorithm is at most $\ln(\delta+1)\mathcal{OPT}$. For each node u, the probability of not being covered after step 2 can be bounded by

$$\Pr[u \text{ uncovered}] \;=\; \left(1-\frac{\ln(\delta+1)}{\delta+1}\right)^{\delta+1} \underset{\text{Fact 1.5}}{\leq} \left(\frac{1}{e}\right)^{\ln(\delta+1)} \;=\; \frac{1}{\delta+1}.$$

Thus, the expected number of nodes joining in Step 3 is at most $\mathcal{OPT}$, and Inequality (2.1) follows.

In the given algorithm, symmetries are broken by choosing the set S at random. In the case of regular graphs, each node can join S with the same probability. If the random decisions are done independently, we can expect to obtain a reasonably good dominating set. If we want to extend the described method to general graphs, we might come up with the following algorithm.

1. For each node v_i, compute a probability p_i.
2. Node v_i joins S with probability p_i.
3. If a node is not covered, it joins S.

This brings up a new problem. How can we determine the probabilities in step 1 such that steps 2 and 3 have the same effect as in regular graphs? It turns out that we have to solve step 1 such that the probabilities of a node u and its neighbors $\Gamma(u)$ sum up to at least $\ln\Delta$ and such that the sum over all probabilities is minimized (Δ is the maximum degree of G). The problem of finding such an optimal probability assignment is known as the *minimum fractional dominating set* (MFDS) problem. To understand the relation between MDS and MFDS we look at a common formulation of MDS as an integer linear program. Assume that the nodes of G are named $v_1,\ldots,v_n$. We assign an indicator variable $x_i \in \{0,1\}$ to each node v_i, such that $x_i = 1 \Leftrightarrow v_i \in S$. For S to be a dominating set, we have to demand that for each node $v_i \in V$ at least one of the nodes in $\Gamma^+(v)$ is in S. Therefore, S is a dominating set of G if and only if $\forall i \in [1,n] : \sum_{j\in\Gamma^+(v)} x_j \geq 1$. The MDS problem can then be formulated as follows:

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} x_i \\ \text{subject to} \quad & N\cdot\underline{x} \geq \underline{1} \\ & x_i \in \{0,1\}. \end{aligned} \tag{$\mathrm{IP}_{\mathrm{DS}}$}$$

Here, the neighborhood matrix N is the adjacency matrix of G with ones in the main diagonal. That is, for each matrix element n_{ij} we have $n_{ij} = 1$ if $(v_i, v_j) \in E \vee i = j$ and $n_{ij} = 0$ otherwise. From ($\mathrm{IP_{DS}}$), we obtain the MFDS problem by allowing the x-variables to take on arbitrary values between 0 and 1, that is, MFDS is the natural LP relaxation of MDS:

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} x_i \\ \text{subject to} \quad & N \cdot \underline{x} \geq \underline{1} \\ & x_i \geq 0. \end{aligned} \tag{$\mathrm{LP_{DS}}$}$$

We have therefore reduced the problem of finding a distributed algorithm for MDS to finding a distributed solution for MFDS. A dominating set can then be computed using a simple randomized rounding scheme. The advantage of this approach is that breaking symmetries can be avoided up to the last step. Based on a solution of the linear program, symmetry breaking becomes a lot simpler. In a certain sense, distributed LP relaxation allows for the separation of the two given tasks, namely solving the dominating set problem and breaking symmetries.

2.1.3 Problem Definition

If we generalize the MDS problem given by ($\mathrm{IP_{DS}}$), we obtain the following integer program:

$$\begin{aligned} \min \quad & \underline{c}^{\mathrm{T}} \underline{x} \\ \text{subject to} \quad & A \cdot \underline{x} \geq \underline{b} \\ & x_i \in \mathbb{N}_0. \end{aligned} \tag{IP}$$

We assume that all entries a_{ij} of A are non-negative and that all entries b_i of $\underline{b}$, and c_i of $\underline{c}$ are positive. Under these assumptions, (IP) is called a covering problem. Many important problems such as (weighted) minimum set cover (MSC), (weighted) minimum dominating set, or (weighted) minimum vertex cover (MVC) fall into this category. A vertex cover is a subset of the nodes which covers all edges. As argued in Section 2.1.2, for a distributed algorithm, it is reasonable to first solve an LP relaxation of (IP) instead of directly solving (IP). In the case of (IP), we obtain a linear program (PP) describing fractional covering problems:

$$\begin{aligned} \min \quad & \underline{c}^{\mathrm{T}} \underline{x} \\ \text{subject to} \quad & A \cdot \underline{x} \geq \underline{b} \\ & x_i \geq 0. \end{aligned} \tag{PP}$$

The dual linear program (DP) is called a fractional packing problem:

$$\begin{aligned} \max \quad & \underline{b}^{\mathrm{T}}\underline{y} \\ \text{subject to} \quad & A^{\mathrm{T}} \cdot \underline{y} \leq \underline{c} \\ & y_i \geq 0. \end{aligned} \tag{DP}$$

Throughout the thesis, (PP) is called the primal LP and (DP) is called the dual LP. Packing problems occur in a broad range of resource allocation problems. As an example, in [17] and [110], the problem of assigning flows to a given fixed set of paths is described. In general, optimal assignments of complex combinations of resources such as bandwidth on given paths, computing cycles on some machines, storage, etc. can be modeled by packing linear programs [110]. Another common packing problem is (weighted) maximum matching, the problem of finding a largest possible set of pairwise non-adjacent edges. Note that maximum matching is a combinatorial problem which requires an integer solution. Analogously to covering problems, integer packing problems are obtained by replacing $y_i \geq 0$ with $y_i \in \mathbb{N}_0$ in (DP). The fractional version of maximum matching is the dual of the fractional minimum vertex cover problem.

While computing a dominating set of the network graph is an inherently distributed task, the problems described by the (integer) linear programs (IP),(PP), and (DP) have no immediate distributed meaning. We therefore need to define a distributed version of these problems, that is, we need a network graph on which we can solve the described linear programs. We use a natural definition which was introduced in [110] and also applied in [17]. For each primal variable x_i and for each dual variable y_j, there are nodes v_i^p and v_j^d, respectively. We denote the set of primal variables by V_p and the set of dual variables by V_d. The network graph $G = (V_p \cup V_d, E)$ is a bipartite graph with the edge set

$$E \;:=\; \left\{(v_i^p, v_j^d) \in V_p \times V_d \,\middle|\, a_{ji} \neq 0\right\},$$

where a_{ji} is the entry of row j and column i of A. We define $m := |V_p|$ and $n := |V_d|$, that is, A is a $(n \times m)$-matrix. Further, the maximum primal and dual degrees are denoted by Δ_p and Δ_d, respectively.

In most real-world examples of distributed covering and packing problems, the network graph is of course not equal to the described bipartite graph. However, it is usually straightforward to simulate an algorithm which is designed for G as above on the actual network graph G'. We briefly describe MDS, MVC and MM as examples. In MDS, A is essentially the adjacency matrix of the network graph G'. Therefore, there is a primal variable x_i and a dual variable y_i for each node $v_i \in V(G')$. The matrix entry a_{ji} is not 0 if and only if $i = j$ or if v_i and v_j are adjacent. Thus, node v_i can simulate v_i^p and v_i^d; all edges of G are then also edges in G'. In MVC and in MM, each primal variable corresponds to a node of G' and each dual variable corresponds to an edge of G'.

The matrix element a_{ji} is non-zero if and only if node v_i is adjacent to edge e_j. Hence, if each dual node v_j^d (edge e_j) is simulated by an adjacent node in G', nodes simulating adjacent nodes in G are also adjacent in G'.

2.1.4 Related Work

Linear programming in general and the use of linear programming techniques in approximation algorithms for combinatorial problems has a long and fruitful tradition in the standard non-distributed setting. For an introduction, we refer to standard text books such as [32] (linear programming) or [65, 127] (approximation algorithms). We have seen that covering and packing problems occur in a variety of applications. It is therefore no surprise that there has been a considerable effort to find specific algorithms for this class of LPs [37, 46, 116] as well as for more general mixed packing and covering LPs [47, 75, 135]. In the context of integer covering and packing problems, the technique of randomized rounding has been introduced to convert fractional solutions into integer ones [104, 117, 124].

The first algorithm which solves covering and packing LPs in a parallel model has been given by Luby and Nisan in [97]. A parallel algorithm for mixed packing and covering problems has been given in [135]. In both algorithms, processors need information about the global state of the system. The problem of approximating positive LPs using only local information has been introduced in [110]. The first algorithm achieving a constant approximation in poly-logarithmic time is described in [17]. We compare our algorithm to the algorithm of [17] at the end of Section 3.2.4.

There is a lot of work on solving specific combinatorial covering and packing problems in a distributed model. This is especially true for the minimum dominating set problem. In a standard non-distributed setting, the complexity of MDS is essentially known. MDS is one of the first problems which was shown to be NP-hard [53, 73]. Using a simple greedy algorithm, it can be approximated by a factor of $\ln \Delta$ [31, 71, 123]. Unless NP $\subseteq$ DTIME($n^{\mathrm{O}(\log\log n)}$), this was shown to be optimal [42]. In [86], a distributed algorithm which computes a dominating set of size at most $n/2$ in $\mathrm{O}(\log^* n)$ rounds has been presented. The given algorithm is extremely fast, the approximation ratio can however be as bad as $\Theta(\Delta)$. The paper [133] provides a constant-time connected dominating set algorithm without any guarantees at all. The first algorithms to achieve a non-trivial approximation ratio are given in [69, 118]. They both achieve an $\mathrm{O}(\log \Delta)$-approximation in poly-logarithmic time. Note that only the algorithm of [69] is explicitly formulated for a distributed communication model. Up to constant factors, they achieve the same approximation ratio as the greedy algorithm. If all local computations are polynomial, we therefore cannot hope to obtain significantly better approximation ratios.

A combinatorial packing problem for which distributed approximation algorithms have been presented is maximum matching. In $\mathrm{O}(\log n)$ time, the maximal matching algorithms of [2, 68, 96] provide 2-approximations for the

maximum matching problem. In [130], a distributed algorithm which computes a constant approximation for the weighted maximum matching problem in $O(\log^2 n)$ rounds is presented.

2.2 Distributed Greedy Dominating Set Algorithm

In this section, we develop an algorithm for the MDS linear program (LP_{DS}) as a first step towards our $\mathcal{CONGEST}_{BC}$ algorithm for general covering and packing linear programs. The achieved time-approximation trade-off of the algorithm of this section is substantially worse than the trade-off of the algorithm which we present in the next section. However, the algorithm of this section is a lot simpler and easier to understand than the improved general algorithm of Section 2.3. Nevertheless, the main ideas underlying both algorithms are the same, that is, Section 2.2 serves as an introduction and should give intuition for Section 2.3.

2.2.1 The Greedy Algorithm

The design and analysis of our distributed covering and packing algorithms are closely related to the greedy dominating set algorithm. We therefore quickly review the greedy algorithm for MDS before describing the distributed algorithms. The greedy MDS algorithm starts with an empty set S. It adds nodes to S as long as S is not a dominating set of the given graph G. In accordance with other dominating set papers (e.g. [58, 69]), we call uncovered nodes white and covered nodes gray, that is, a node v is gray as soon as v or a neighbor of v is in S. In each step, the greedy algorithm picks the node u with the maximum number of white nodes in its closed neighborhood $\Gamma^+(u)$ and adds u to S. The following Theorem 2.1 is well-known [31, 71, 123]; we give a proof in order to show the main technique (dual fitting) for the analysis of our distributed algorithms.

The method of dual fitting can be described as follows. While computing a primal solution (in our case a dominating set), we also construct a solution to the dual LP. The dual solution is constructed such that the primal solution is fully paid for by the dual solution, that is, assuming the primal problem to be a minimization problem, the value of the dual objective function is always at least the value of the primal objective function. If the computed solution is not optimal, the dual solution is infeasible. However, if each inequality of the dual LP is satisfied up to a factor of α, dividing all dual variables by α makes the dual solution feasible. By LP duality, the obtained primal solution is then greater than an optimal solution by a factor of at least α, that is, the computed solution is an α-approximation.

Theorem 2.1. *The greedy MDS algorithm computes a dominating set S which is greater than an optimal dominating set at most by a factor of $H(\Delta+1) < \ln(\Delta+1)+1$, where $H(q)$ denotes the q^{th} harmonic number.*

Proof. At the beginning, $S = \emptyset$ and hence also $\forall i : x_i = y_i = 0$. Every time we add a node u to S, the objective function of ($\mathrm{LP}_{\mathrm{DS}}$) is incremented by 1. Following the described approach, we also have to increase the objective function of ($\mathrm{DP}_{\mathrm{DS}}$) by 1. We do this by equally charging all nodes v which have become gray when we added u to S. Hence, if there are m white nodes in $\Gamma^+(u)$, the dual variable of each of those nodes is increased by $1/m$. Note that only dual variables of white nodes becoming gray are increased. Thus, every dual variable is increased exactly once.

Let us now consider an inequality of the dual LP ($\mathrm{DP}_{\mathrm{DS}}$). The sum of all y-values in the closed neighborhood $\Gamma^+(u)$ of a node u has to be at most 1. How large can the sum of y-values in $\Gamma^+(u)$ become? At the beginning, all neighbors of u are white. Assume that the nodes of $\Gamma^+(u)$ become gray in t steps such that in step i, a_i nodes of $\Gamma^+(u)$ become gray. We have $\sum_{i=1}^{t} a_i = \delta$, where $\delta := |\Gamma^+(u)|$. Let $\delta_i := \delta - \sum_{i=1}^{i-1} a_i$ be the number of white nodes before the i^{th} step. By adding u to S after step $i-1$, it would be possible to color δ_i nodes gray. Hence, the greedy algorithm selects a node which covers at least δ_i nodes and thus the y-values of step i can be at most $1/\delta_i$. We now get

$$\sum_{v_i \in \Gamma^+(u)} y_i \;\leq\; \sum_{i=1}^{t} \frac{a_i}{\delta_i} \;=\; \sum_{i=1}^{t} \sum_{j=1}^{a_i} \frac{1}{\delta_i} \;\leq\; \sum_{i=0}^{\delta-1} \frac{1}{\delta - i} \;=\; H(\delta) \leq H(\Delta + 1),$$

which completes the proof. □

2.2.2 Basic Algorithm: Nodes Know Maximum Degree

We now show how to turn the described greedy MDS algorithm into a distributed algorithm for ($\mathrm{LP}_{\mathrm{DS}}$), the LP relaxation of MDS. This way, we achieve an $\mathrm{O}(k\Delta^{\mathrm{O}(1)/k})$-approximation of ($\mathrm{LP}_{\mathrm{DS}}$) in $\mathrm{O}(k^2)$ rounds. For the sake of simplicity and clarity, we first present an algorithm which assumes that all nodes know the largest degree Δ of the network (Section 2.2.2). In a second step (Section 2.2.3), we describe how Δ can be approximated locally such that the global knowledge of Δ is not necessary any more.

The main problem of applying the greedy algorithm in a distributed environment is the synchronization of different nodes capable of joining the dominating set. Thus, we have to solve a classical symmetry breaking problem. In Section 2.1.2, we have argued that a simple way to avoid symmetry breaking is to solve ($\mathrm{LP}_{\mathrm{DS}}$) instead of ($\mathrm{IP}_{\mathrm{DS}}$). Intuitively, this means that whenever there are q neighbors of a node u which could all join the dominating set according to the greedy condition, instead of selecting one of the q nodes, each of them increases its x-value by $1/q$.

During the algorithm's execution, all nodes v_i start with $x_i = 0$ and increase their x-values over time. Analogously to Section 2.2.1, we say that a node v_i is colored gray as soon as the sum of the weights x_j for $v_j \in \Gamma^+(v_i)$ exceeds 1, that is, as soon as the node is completely covered. Initially all nodes are colored white. The number of white nodes $v_j \in \Gamma^+(v_i)$ at a given time is called the

Algorithm 1 $\mathrm{LP}_{\mathrm{DS}}$ approximation (Δ known)

1: $x_i := 0$; $\tilde{\delta}(v_i) := \delta(v_i) + 1$; $\mathrm{color}_i :=$ 'white';
2: **for** $\ell := k-1$ to 0 by -1 **do**
3: $\quad(*\ \tilde{\delta}(v_i) \leq (\Delta+1)^{(\ell+1)/k},\ y_i := 0\ *)$
4: $\quad$**for** $m := k-1$ to 0 by -1 **do**
5: $\quad\quad(*\ a(v_i) \leq (\Delta+1)^{(m+1)/k}\ *)$
6: $\quad\quad$**if** $\tilde{\delta}(v_i) \geq (\Delta+1)^{\ell/k}$ **then**
7: $\quad\quad\quad x_i := \max\left\{x_i, \frac{1}{(\Delta+1)^{m/k}}\right\}$
8: $\quad\quad$**fi**;
9: $\quad\quad$**send** x_i to all neighbors;
10: $\quad\quad$**if** $\sum_{j\in\Gamma^+(v_i)} x_j \geq 1$ **then** $\mathrm{color}_i :=$ 'gray' **fi**;
11: $\quad\quad$**send** color_i to all neighbors;
12: $\quad\quad\tilde{\delta}(v_i) := \left|\{j \in \Gamma^+(v_i) \mid \mathrm{color}_j = \text{'white'}\}\right|$
13: $\quad$**od**
14: $\quad(*\ y_i \leq 1/(\Delta+1)^{(\ell-1)/k}\ *)$
15: **od**

dynamic degree of v_i and denoted by $\tilde{\delta}(v_i)$. When starting the algorithm, all nodes are white and thus $\tilde{\delta}(v_i) = \delta(v_i) + 1$.

Assume now that all nodes know Δ, the maximum degree of the network. Algorithm 1 is synchronously executed by all nodes. Before coming to a detailed analysis of Algorithm 1, we give a general overview. Each node v_i calculates the corresponding component x_i of the solution for $\mathrm{LP}_{\mathrm{DS}}$. Compared to the sequential greedy algorithm where in each step exactly one node increases its x-value from 0 to 1, Algorithm 1 raises the x-values of many nodes simultaneously. In order to avoid the problem of overloading a node which has many neighbors increasing their x_i, we only increase the x-values by small amounts each time. Initially all x_i are set to 0; they are gradually increased as the algorithm progresses. The algorithm consists of two nested loops. The purpose of the outer loop is to reduce the highest dynamic degree in the network. As indicated by the invariant in Line 3, $\tilde{\delta}(v_i)$ is reduced by a factor of $(\Delta+1)^{1/k}$ in every iteration of the outer loop. Each iteration of the outer loop yields an infeasible primal solution. By rearranging the weights in a similar way as in the original greedy set cover proof, this primal solution can be converted into a dual solution $\underline{y} = (y_1, \ldots, y_n)$ which is feasible up to a factor of $(\Delta+1)^{2/k}$. The combined primal solutions of all outer loop iterations give a primal feasible $k(\Delta+1)^{2/k}$-approximation.

As in the sequential greedy algorithm, only the nodes of large degree increase their x-values (Line 7). In the inner loop, the x-values are increased stepwise. We call the high-degree nodes which increase their x-values in Line 7 active nodes. Let $a(v_j)$ be the number of active neighbors of v_j. The increase of variable x_i of a node v_i is at most indirectly proportional to the maximum $a(v_j)$ among all white neighbors v_j of v_i. By this, the maximum number of

active neighbors $a(v)$ among all nodes v is reduced in every iteration of the inner loop (invariant in Line 5) and we can guarantee that the total x-increase is not too high. That is, no node is overloaded because of too many neighbors increasing their x_i by too much. Lemma 2.2 explains the invariant of Line 3.

Lemma 2.2. *At the beginning of each iteration ℓ of the outer loop of Algorithm 1, that is, at Line 3, the dynamic degree $\tilde{\delta}(v_i)$ of each node v_i is*

$$\tilde{\delta}(v_i) \leq (\Delta+1)^{(\ell+1)/k}.$$

Proof. For $\ell = k-1$ the condition reduces to $\tilde{\delta}(v_i) \leq \Delta+1$ and therefore follows from the definition of Δ. For all other iterations the lemma is true because in the very last step of the preceding iteration $(\ell+1)$, all nodes with $\tilde{\delta}(v_i) \geq (\Delta+1)^{(\ell+1)/k}$ set $x_i := 1$ in Line 7. By this all nodes in $\Gamma^+(v_i)$ turn gray and therefore $\tilde{\delta}(v_i)$ becomes 0. Thus all nodes for which the condition of the lemma does not hold set $x_i := 1$ and therefore $\tilde{\delta}(v_i)$ is set to 0. □

In a single iteration of the outer loop, only active nodes, that is, nodes with $\tilde{\delta}(v_i) \geq (\Delta+1)^{\ell/k}$ increase their x-value (Lines 6-8). The number of active nodes in the closed neighborhood $\Gamma^+(v_i)$ of a white node v_i at the beginning of an inner-loop iteration (Line 5) is called $a(v_i)$. We define $a(v_i)$ to be 0 if v_i is gray. The purpose of the inner loop is to gradually reduce the maximum $a(v)$ in the graph (invariant in Line 5).

Lemma 2.3. *At the beginning of each iteration of the inner loop of Algorithm 1, that is at Line 5,*

$$a(v_i) \leq (\Delta+1)^{(m+1)/k}$$

for all nodes $v_i \in V$.

Proof. For $m = k-1$ we have $a(v_i) \leq \Delta+1$ which is always true. For $m < k-1$, we prove that all nodes v_i with more than $(\Delta+1)^{(m+1)/k}$ active neighbors are gray and therefore $a(v_i) = 0$. It is sufficient to show that all nodes v_i for which $a(v_i) > (\Delta+1)^{m/k}$ in Line 5 are colored gray at the end of the inner-loop iteration (i.e., after Line 14). All active nodes v_j increase x_j such that $x_j \geq 1/(\Delta+1)^{m/k}$ (Lines 6-8 of Algorithm 1). If $a(v_i) > (\Delta+1)^{m/k}$, there are more than $(\Delta+1)^{m/k}$ active nodes in $\Gamma^+(v_i)$. Therefore the sum of the x-values in $\Gamma^+(v_i)$ is at least 1 after Line 10. Figure 2.1 serves as an illustration for Lemma 2.3. □

In order to bound the weights assigned during the iterations of the inner loop, we assign a dual variable y_i to each node v_i. In Line 3 all y_i are set to 0. Whenever a node v_i increases x_i, the additional weight is equally distributed among the y_j of all the nodes v_j in $\Gamma^+(v_i)$ which were white before the increase of x_i as in the analysis of the greedy algorithm (Theorem 2.1). Hence the sum of the y-values is always equal to the sum of the x-increases during the current iteration of the outer loop. In Lemma 2.4, we show that at the end of every iteration of the outer loop, that is at Line 14, all y_i are bounded by roughly the

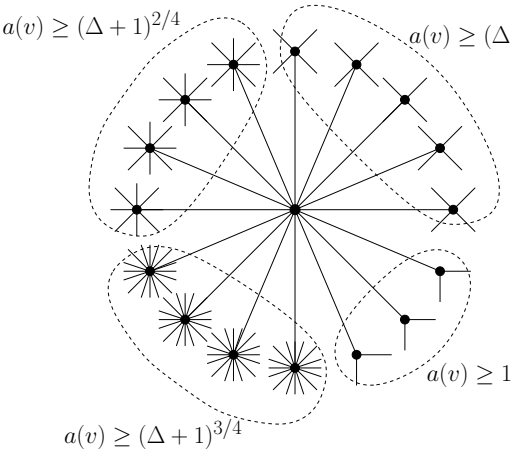

Figure 2.1: Example with $k = 4$: First, the nodes which have $a(v) \geq (\Delta+1)^{3/4}$ active neighbors are covered when the x-values are set to $1/(\Delta+1)^{3/4}$, then the nodes which have $a(v) \geq (\Delta+1)^{2/4}$ active neighbors are covered when the x-values are set to $1/(\Delta+1)^{2/4}$, and so on. By this, it is guaranteed that the dual weights do not become too large.

reciprocal of the dynamic degree of active nodes. Together with the invariant in Line 3 (Lemma 2.2), this enables us to prove a bound on the total weight of the additional x-values in each iteration of the outer loop.

Lemma 2.4. *At the end of an iteration of the outer loop of Algorithm 1, that is at Line 14,*

$$y_i \leq \frac{1}{(\Delta+1)^{\frac{\ell-1}{k}}}$$

for all nodes $v_i \in V$.

Proof. Because y_i is set to 0 in Line 3, we only have to consider a single iteration of the outer loop (ℓ-loop), that is a period in which ℓ remains constant. The value of y_i can only be increased as long as v_i is a white node. The increments all happen in Line 7 because the x-values are increased only there. For each white node v_i, we divide the iteration of the outer loop into two phases. The first phase consists of all inner-loop iterations where v_i remains white. The second phase consist of the remaining inner-loop iterations where v_i becomes or is gray. Assume first that v_i remains white after the first iteration of the inner-loop, that is, the first phase exists. During the whole first phase, we have $\sum_{j\in\Gamma^+(v_i)} x_j < 1$. Because all increases of x-values are distributed among at

least $(\Delta+1)^{\ell/k}$ y-values, we therefore get

$$y_i \;\leq\; \frac{\sum_{j\in\Gamma^+(v_i)} x_j}{(\Delta+1)^{\frac{\ell}{k}}} \;<\; \frac{1}{(\Delta+1)^{\frac{\ell}{k}}} \tag{2.2}$$

for phase 1. In Line 7 of the first inner-loop iteration of the second phase, y_i gets its final value because only y-values of white nodes are increased. All active nodes have already been active in the preceding inner-loop iteration because $\tilde{\delta}(v_j)$ can only become smaller over time. Thus from the preceding iteration, all $a(v_i)$ active nodes $v_j \in \Gamma^+(v_i)$ have $x_j \geq 1/(\Delta+1)^{(m+1)/k}$. In Line 7, the x-values of these nodes are now increased to $1/(\Delta+1)^{m/k}$. The difference of this value is distributed among at least $(\Delta+1)^{\ell/k}$ y-values and so the increase of y_i is at most

$$\frac{\frac{1}{(\Delta+1)^{\frac{m}{k}}} - \frac{1}{(\Delta+1)^{\frac{m+1}{k}}}}{(\Delta+1)^{\frac{\ell}{k}}} a(v_i). \tag{2.3}$$

To obtain a bound on y_i, we have to add its value before the increase, which is given by Inequality (2.2). From Lemma 2.3 we know that $a(v_i) \leq (\Delta+1)^{(m+1)/k}$. Plugging this into the sum of Inequalities (2.2) and (2.3), we obtain

$$y_i \leq \frac{(\Delta+1)^{\frac{1}{k}}-1}{(\Delta+1)^{\frac{\ell}{k}}} + \frac{1}{(\Delta+1)^{\frac{\ell}{k}}} = \frac{1}{(\Delta+1)^{\frac{\ell-1}{k}}}.$$

If v_i becomes gray in the first inner-loop iteration, we have

$$y_i \;\leq\; \frac{\delta(v_i)+1}{(\Delta+1)^{(k-1)/k}(\Delta+1)^{\ell/k}} \;\leq\; \frac{\Delta+1}{(\Delta+1)^{(k+\ell-1)/k}} \;=\; \frac{(\Delta+1)^{1/k}}{(\Delta+1)^{\ell/k}}.$$

□

Based on the given lemmas, we can now look at the overall approximation ratio of Algorithm 1.

Theorem 2.5. *For all network graphs G, Algorithm 1 computes a feasible solution x for the linear program $\mathrm{LP_{DS}}$ such that x is a $k(\Delta+1)^{2/k}$-approximation of $\mathrm{LP_{DS}}$. Further, Algorithm 1 terminates after $2k^2$ rounds.*

Proof. For the number of rounds, observe that each iteration of the inner loop involves the sending of two messages and therefore takes two rounds. The number of such iterations is k^2.

Further, the calculated x-values form a feasible solution of $\mathrm{LP_{DS}}$ because in the very last iteration of the inner loop ($\ell = 0$, $m = 0$) all nodes v_i with $\tilde{\delta}(v_i) \geq 1$ set $x_i := 1$. This includes all remaining white nodes. We prove the approximation ratio of $k(\Delta+1)^{2/k}$ by showing that the additional weight (i.e., sum of x-values) is upper-bounded by $(\Delta+1)^{2/k}$ in each iteration of the outer loop. From Lemma 2.2, we know that at Line 3, that is, when the iteration starts, the dynamic degree $\tilde{\delta}(v_i)$ of each node v_i is $\tilde{\delta}(v_i) \leq (\Delta+1)^{(\ell+1)/k}$. Hence

there are at most $(\Delta+1)^{(\ell+1)/k}$ non-zero y-values in the closed neighborhood of every node v_i at the end of an outer-loop iteration at Line 14. Furthermore, Lemma 2.4 implies that all y-values are at most $(\Delta+1)^{-(\ell-1)/k}$ at Line 14. The sum of the y-values in the neighborhood of a node v_i during each iteration of the outer loop is therefore upper-bounded by

$$\sum_{j\in\Gamma^+(v_i)} y_j \leq \frac{(\Delta+1)^{\frac{\ell+1}{k}}}{(\Delta+1)^{\frac{\ell-1}{k}}} = (\Delta+1)^{\frac{2}{k}}.$$

If we divide each y-value by $(\Delta+1)^{2/k}$ ($y'_i := y_i/(\Delta+1)^{2/k}$), the new y-values form a feasible solution for the dual LP ($\mathrm{DP}_{\mathrm{DS}}$) because $\forall i : \sum_{j\in\Gamma^+(v_i)} y'_j \leq 1$. Hence the sum of all new y-values is a lower bound on the size of an optimal dominating set $\mathcal{OPT}$; therefore

$$\sum_{i=1}^{n} y_i \leq (\Delta+1)^{2/k}|\mathcal{OPT}|$$

for every iteration of the outer loop. Because $\underline{y}$ is defined such that the sum over all y-values is equal to the sum over all increases of the x-values, and because there are k iterations of the outer loop, we have

$$\sum_{i=1}^{n} x_i \leq k(\Delta+1)^{2/k}|\mathcal{OPT}|$$

at the end of Algorithm 1. □

2.2.3 Improved Algorithm: No Global Knowledge

The only thing which cannot be calculated locally in Algorithm 1 is the maximum degree Δ. In this section, we show how each node v can locally compute an estimate $\widehat{\Delta}(v)$ for Δ such that v can replace each occurrence of Δ by $\widehat{\Delta}(v)$ in Algorithm 1. A straightforward way to estimate Δ would be to let $\widehat{\Delta}(v)$ be the degree $\delta(v)$ of v. More generally, for a fixed t $\widehat{\Delta}(v)$ could be defined as the largest degree at distance at most t from v. Those simple estimates have one major problem. The gap between estimates $\widehat{\Delta}(u)$ and $\widehat{\Delta}(v)$ for adjacent nodes u and v can be arbitrarily large. However, the analysis of Algorithm 1 relies on the fact that neighboring nodes have at least a similar notion of Δ.

Nevertheless, it can be shown that it is possible to estimate Δ by essentially taking the maximum degree in distance at most 2. In [81], we estimate the maximum dynamic degree in this way. However, the nodes have to compute a new estimate in every iteration of the outer for-loop. Here, we present a more general approach. At the beginning, each node v_i computes an estimate $\widehat{\Delta}(v_i)$. Apart from substituting $\widehat{\Delta}(v_i)$ for Δ, we do not need to change anything in Algorithm 1.

Algorithm 2 Computing local estimates of a global maximum

1: $\widehat{Q}(v_i) := q_i$; $s_i := q_i^{(k-1)/k}$;
2: **for** $\ell := k-2$ **to** 1 **by** -1 **do**
3: **send** s_i to all neighbors $\Gamma(v_i)$;
4: $t_i := \max_{v_j \in \Gamma(v_i)} s_j$;
5: **if** $t_i > \widehat{Q}(v_i)$ **then** $\widehat{Q}(v_i) := t_i$ **fi**;
6: $s_i := t_i^{\ell/(\ell+1)}$
7: **od**

To estimate Δ, we look at the following problem. Assume that we are given a network graph $G = (V, E)$. Each node $v_i \in V$ of G has some local value $q_i \geq 1$. Let $Q := \max_i q_i$ be the largest q_i in the graph. Using a distributed algorithm, every node v_i wants to compute an estimate $\widehat{Q}(v_i)$ of Q such that $q_i \leq \widehat{Q}(v_i) \leq Q$ and such that for adjacent nodes v_i and v_j, the estimates differ by a factor of at most $Q^{1/k}$ for an arbitrary integer $k \geq 1$, that is,

$$(v_i, v_j) \in E \implies \frac{1}{Q^{1/k}} \leq \frac{\widehat{Q}(v_i)}{\widehat{Q}(v_j)} \leq Q^{1/k}. \tag{2.4}$$

Algorithm 2 shows how to compute estimates $\widehat{Q}(v_i)$ satisfying these conditions. Let $d_G(u, v)$ be the distance between nodes u and v on G. The algorithm computes

$$\widehat{Q}(v_i) := \max_j \left(q_j^{\frac{k-d_G(v_i,v_j)}{k}} \right) \tag{2.5}$$

in $k-1$ rounds. The following theorem proves this and shows that these estimates in fact fulfill the given conditions.

Theorem 2.6. *Algorithm 2 computes estimates $\widehat{Q}(v_i)$ for Q as given by Equation* (2.5)*. The estimates satisfy $q_i \leq \widehat{Q}(v_i) \leq Q$ as well as Inequality* (2.4)*. Algorithm 2 terminates after $k-1$ rounds.*

Proof. In each iteration of the for-loop, one message is sent by every node. Because there are $k-1$ iterations, the time complexity of the algorithm is $k-1$. It remains to show $q_i \leq \widehat{Q}(v_i) \leq Q$. At the beginning $\widehat{Q}(v_i) = q_i$. If $\widehat{Q}(v_i)$ is set to t_i in Line 5, it is because $t_i > \widehat{Q}(v_i)$ and hence $\widehat{Q}(v_i) \geq q_i$. Since each s_i and hence also each t_i is smaller than q_j for some j, we also have $\widehat{Q}(v_i) \leq Q$. It remains to prove that the computed estimates satisfy Equation (2.5) and therefore Inequality (2.4).

We first prove that Algorithm 2 really computes estimates as given by Equation (2.5) and then show that this implies Inequality (2.4). Let us call the for-loop iteration for which $\ell = k-1-r$ the r^{th} iteration. Let $t_i(r)$ be the value t_i computed in the r^{th} iteration. We claim that

$$t_i(r) = \max_{v_j \in \Gamma_r(v_i)} \left(q_j^{\frac{k-r}{r}} \right), \tag{2.6}$$

where $\Gamma_r(v_i)$ is the set of all nodes v_j with $d_G(v_i,v_j) \leq r$. From this, Equation (2.5) follows. Note that if $d_G(v_i,v_j) \geq k$, $q_j^{(k-r)/k} \leq 1 \leq q_i \leq \widehat{Q}(v_i)$. Equation (2.6) can be shown by induction on r. For $r = 1$, Equation (2.6) is clearly true. For $r > 1$, we have

$$\begin{aligned} t_i(r) &= \max_{v_j \in \Gamma(v_i)} \left(t_j (r-1)^{\frac{\ell+1}{\ell+2}} \right) = \max_{v_j \in \Gamma(v_i)} \left(t_j (r-1)^{\frac{k-r}{k-r+1}} \right) \\ &= \max_{v_j \in \Gamma v_i} \left(\max_{v_h \in \Gamma_{r-1}(v_j)} \left(q_h^{\frac{k-r+1}{k}} \right)^{\frac{k-r}{k-r+1}} \right) = \max_{v_j \in \Gamma_r(v_i)} \left(q_j^{\frac{k-r}{r}} \right). \end{aligned}$$

It remains to be proved that Inequality (2.4) follows from Equation (2.5). For the sake of contradiction, assume that there are adjacent nodes v_i and v_j for which $\widehat{Q}(v_i)/\widehat{Q}(v_j) > Q^{1/k}$. By Equation (2.5), there is some h for which $\widehat{Q}(v_i) = q_h^{(k-d_G(v_i,v_h))/k}$. However, this means that $\widehat{Q}(v_j) \geq q_h^{(k-d_G(v_i,v_h)-1)/k} = \widehat{Q}(v_i)/q_h^{1/k}$. This is a contradiction because $q_h \leq Q$. □

We now assume that all nodes first apply Algorithm 2 and afterwards Algorithm 1 to compute a solution for ($\mathrm{LP_{DS}}$). In Algorithm 1, every time Δ occurs, it is substituted by $\widehat{Q}(v_i)$. Both algorithms have a parameter k which can be chosen arbitrarily. From here on, we denote them by k_Δ for Algorithm 2 and by k for Algorithm 1. For simplicity, we introduce $\alpha_\Delta := \Delta^{1/k_\Delta}$ for the maximal ratio between the Δ-estimates of adjacent nodes. Lemma 2.7 is analogous to Lemma 2.2.

Lemma 2.7. *At the beginning of each iteration ℓ of the outer loop of Algorithm 1, that is at Line 3, the dynamic degree $\tilde{\delta}(v_i)$ of each node v_i is*

$$\tilde{\delta}(v_i) \leq (\widehat{\Delta}(v_i)+1)^{(\ell+1)/k}.$$

Proof. Same proof as for Lemma 2.2. □

The following lemma corresponds to a combination of Lemmas 2.3 and 2.4.

Lemma 2.8. *At the end of an iteration of the outer loop of Algorithm 1, that is at Line 14,*

$$y_i \leq \frac{\alpha_\Delta (\Delta+1)^{1/k}}{(\widehat{\Delta}(v_i)+1)^{\ell/k}}$$

for all nodes $v_i \in V$.

Proof. We again look at a single iteration of the outer loop and look at the same two phases as in the proof of Lemma 2.4 (inner-loop iterations where v_i remains white and inner-loop iterations where v_i becomes or is gray). If v_i becomes gray in the first inner-loop iteration, we have

$$y_i \leq \sum_{v_j \in \Gamma^+(v_i)} \frac{1}{(\widehat{\Delta}(v_j)+1)^{(k+\ell-1)/k}} \leq \frac{\big(\delta(v_i)+1\big)\alpha_\Delta}{(\widehat{\Delta}(v_i)+1)^{(k+\ell-1)/k}} \leq \frac{\alpha_\Delta}{(\widehat{\Delta}(v_i)+1)^{(\ell-1)/k}}$$

in analogy to Lemma 2.4. Now assume that v_i does not become gray in the first inner-loop iteration. Then, the y-values from phase 1 can be bounded as:

$$y_i \leq \sum_{v_j \in \Gamma^+(v_i)} \frac{x_j}{(\widehat{\Delta}(v_j)+1)^{\ell/k}} \leq \frac{\alpha_\Delta}{(\widehat{\Delta}(v_i)+1)^{\ell/k}}. \tag{2.7}$$

Let us now look at the inner-loop iteration where v_i becomes gray. Let A_i denote the set of active nodes in $\Gamma^+(v_i)$ during this m-loop iteration. At the beginning of this inner-loop iteration, we have

$$\forall v_j \in A_i : x_j \geq \frac{1}{(\widehat{\Delta}(v_j)+1)^{(m+1)/k}} \quad \text{and} \quad \sum_{v_j \in A_i} x_j < 1. \tag{2.8}$$

The value of y_i is increased by y_i^+ as follows:

$$\begin{aligned}
y_i^+ &\leq \sum_{j \in A_i} \frac{\frac{1}{(\widehat{\Delta}(v_j)+1)^{m/k}} - \frac{1}{(\widehat{\Delta}(v_j)+1)^{(m+1)/k}}}{(\widehat{\Delta}(v_j)+1)^{\ell/k}} \\
&= \sum_{j \in A_i} \frac{\left(\widehat{\Delta}(v_j)+1\right)^{1/k} - 1}{(\widehat{\Delta}(v_j)+1)^{(m+\ell+1)/k}} \leq \sum_{j \in A_i} \frac{(\Delta+1)^{1/k}-1}{(\widehat{\Delta}(v_j)+1)^{(\ell+m+1)/k}} \\
&= \sum_{j \in A_i} \frac{\frac{(\Delta+1)^{1/k}-1}{(\widehat{\Delta}(v_j)+1)^{\ell/k}}}{(\widehat{\Delta}(v_j)+1)^{(m+1)/k}} < \frac{(\Delta+1)^{1/k}-1}{(\widehat{\Delta}(v_j)+1)^{\ell/k}} \leq \frac{\alpha_\Delta((\Delta+1)^{1/k}-1)}{(\widehat{\Delta}(v_i)+1)^{\ell/k}}.
\end{aligned}$$

The second-to-last inequality follows from Inequality (2.8). Together with Inequality (2.7), this proves the lemma. □

For Algorithm 1 with locally estimated maximum degree, we then obtain the following result.

Theorem 2.9. *If v_i uses $\widehat{\Delta}(v_i)$ as computed by Algorithm 2 with parameter k_Δ as an estimate for Δ, for all network graphs G Algorithm 1 computes a feasible solution $\underline{x}$ for $(\mathrm{LP_{DS}})$ with approximation ratio*

$$k(\Delta+1)^{2/k_\Delta}(\Delta+1)^{2/k}.$$

The time complexity is $k_\Delta + 2k^2$.

Proof. By Lemmas 2.7 and 2.8, the sum of the y-values in $\Gamma(v_i)$ of a single iteration of the outer loop is

$$\begin{aligned}
\sum_{v_j \in \Gamma(v_i)} y_j &\leq \frac{(\widehat{\Delta}(v_i)+1)^{(\ell+1)/k}\alpha_\Delta(\Delta+1)^{1/k}}{(\widehat{\Delta}(v_j)+1)^{\ell/k}} \leq \alpha_\Delta^2(\Delta+1)^{2/k} \\
&= (\Delta+1)^{2/k_\Delta}(\Delta+1)^{1/k}.
\end{aligned}$$

The rest of the proof is analogous to the proof of Theorem 2.5. □

2.3 Approximating General Fractional Covering and Packing Problems

After describing a distributed algorithm for the MDS linear program ($\mathrm{LP}_{\mathrm{DS}}$) in Section 2.2, the goal of this section is to generalize the ideas of Algorithms 1 and 2 to get a distributed approximation scheme for (PP) and (DP). Similar to Algorithm 1, which is based on the greedy dominating set algorithm (Section 2.2.1), the algorithm of this section is based on an extension of the greedy algorithm which is described in the following.

2.3.1 Greedy Fractional Dominating Set Algorithm

The variant of the greedy algorithm discussed in this section has been presented in [37, 134]. Instead of solving the MDS problem, the goal is to solve the corresponding LP ($\mathrm{LP}_{\mathrm{DS}}$). We have seen that this is the only way to break the $\Omega(\log \Delta)$ barrier for MDS unless NP $\subseteq$ DTIME($n^{\mathrm{O}(\log \log n)}$) [42]. How can we modify the greedy algorithm to achieve a constant approximation or to even obtain an approximation scheme for ($\mathrm{LP}_{\mathrm{DS}}$)? The main idea is to cover each node multiple times such that in the end each node is covered logarithmically often and such that the dual LP ($\mathrm{DP}_{\mathrm{DS}}$) is still satisfied up to a logarithmic factor. Dividing all variables by the appropriate values then gives feasible solutions for ($\mathrm{LP}_{\mathrm{DS}}$) and ($\mathrm{DP}_{\mathrm{DS}}$) with objective values differing by a constant factor only.

Each node v_i is assigned a requirement r_i which is initially set to 1. In each step of the algorithm, the x-value of a node v_i with maximum

$$\varphi_i \;:=\; \sum_{v_j \in \Gamma^+(v_i)} r_j$$

is incremented by 1. The requirements of all nodes $v_j \in \Gamma^+(v_i)$ are then divided by some given factor $\alpha > 1$. If we have $r_j = \alpha^{-f}$, we set $r_j := 0$. There, f is a positive integer which we will determine later. Hence, r_j is 0 as soon as node v_j is covered $f \geq 1$ times. The optimal values of α and f will be determined later. As in the case of the greedy MDS algorithm, we construct a dual solution with the same objective value along with the primal solution. If x_i is incremented by 1, for each neighbor $v_j \in \Gamma^+(v_i)$, the dual value y_j is increased by r_j/φ_i. By the definition of φ_i, the sum of the dual increments is 1. Thus the objective values of the primal and dual LPs remain the same in each step. The following lemma gives an upper bound on the sum of the y-values in each neighborhood $\Gamma^+(v_i)$.

Lemma 2.10. *At the end of the greedy fractional dominating set algorithm, that is, when all $r_j = 0$, we have*

$$\forall v_i \in V \;:\; \sum_{v_j \in \Gamma^+(v_i)} y_j \;\leq\; \frac{\alpha}{\alpha - 1}\left(\ln \Delta + f \ln \alpha\right).$$

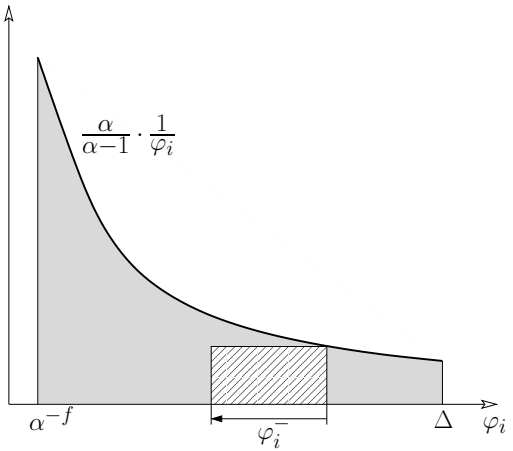

Figure 2.2: Illustration of the increase Y_i^+ of Y_i in a single step of the greedy (PP)-algorithm. If φ_i is reduced by φ_i^-, Y_i is increased by $Y_i^+ \leq \frac{\alpha}{\alpha-1}\frac{\varphi_i^-}{\varphi_i}$. The area of the dashed rectangle is equal this bound on Y_i^+.

Proof. Let $Y_i := \sum_{v_j \in \Gamma^+(v_i)} y_j$. We prove the lemma by analyzing the increase of Y_i as φ_i decreases. We therefore look at a single step in the algorithm where some primal variable x_ℓ of v_ℓ is incremented by 1. Assume that there is a node $v_j \in \Gamma^+(v_i) \cap \Gamma^+(v_\ell)$ which is not yet covered f times. When x_ℓ is increased, the requirement r_j is divided by α or set to 0 and y_j is increased by $y_j^+ := r_j/\varphi_\ell$. Let r_j^- be the decrease of r_j, that is, $r_j^- \geq (1-1/\alpha)r_j$. Because $\varphi_\ell \geq \varphi_i$ (greedy condition), we have

$$y_j^+ \;=\; \frac{r_j}{\varphi_\ell} \;\leq\; \frac{r_j}{\varphi_i} \;\leq\; \frac{\alpha}{\alpha-1} \cdot \frac{r_j^-}{\varphi_i}.$$

By the definitions of Y_i and φ_i, we therefore have

$$Y_i^+ \;\leq\; \frac{\alpha}{\alpha-1} \cdot \frac{1}{\varphi_i} \cdot \varphi_i^- \tag{2.9}$$

where Y_i^+ and φ_i^- are the increase and decrease of Y_i and φ_i, respectively. To understand Inequality (2.9), consider Figure 2.2 where the function $\alpha/(\alpha-1) \cdot 1/\varphi_i$ is plotted against φ_i. If φ_i is decreased by φ_i^-, the area of the dashed rectangle in Figure 2.2 equals the upper bound on Y_i^+ given by Inequality (2.9). As φ_i decreases, the increases Y_i^+ can be visualized as a series of rectangles which all are below the $\alpha/(\alpha-1) \cdot 1/\varphi_i$ curve. The sum of all Y_i^+ is therefore upper-bounded by the area below this curve. As long as $\varphi_i > 0$, it cannot get smaller than α^{-f} because at least one requirement r_j has to be non-zero.

The final value of Y_i is therefore upper bounded by

$$Y_i \;\leq\; \frac{\alpha}{\alpha-1}\int_{\alpha^{-f}}^{\delta_i}\frac{1}{\varphi_i}d\varphi_i \;\leq\; \frac{\alpha}{\alpha-1}\int_{\alpha^{-f}}^{\Delta}\frac{1}{\varphi_i}d\varphi_i \;=\; \frac{\alpha}{\alpha-1}\left(\ln\Delta + f\ln\alpha\right).$$

□

At the end of the algorithm, we divide all primal variables by the largest possible factor keeping the primal solution feasible. All dual variables are divided by the smallest value which makes the dual solution feasible. Because of Lemma 2.10, dividing each dual variable by the given upper bound on Y_i yields a feasible dual solution. Further, the greedy ($\mathrm{LP_{DS}}$)-algorithm is designed such that in the end each node is covered f times. We can therefore divide each primal variable by f and still have a primal feasible solution. Combining both observations, we can bound the ratio ρ between primal and dual objective values and thus the achieved approximation ratio as follows:

$$\rho \;\leq\; \frac{\alpha(\ln\Delta + f\ln\alpha)}{(\alpha-1)f} \;\leq\; \alpha + \frac{\alpha\ln\Delta}{(\alpha-1)f}. \tag{2.10}$$

The second inequality follows from $\ln\alpha \leq \alpha - 1$. For an arbitrary $\varepsilon > 0$, let $\alpha = 1 + \varepsilon/2$ and $f = 4\ln\Delta/\varepsilon^2$. Plugging those values into Inequality (2.10) gives $\rho < 1 + \varepsilon$. We therefore obtain the following theorem.

Theorem 2.11. *Using appropriate values for α and f, for all $\varepsilon > 0$ the described greedy fractional dominating set algorithm computes $(1+\varepsilon)$-approximate solutions for* ($\mathrm{LP_{DS}}$) *and* ($\mathrm{DP_{DS}}$). *For constant ε, $\alpha \in \mathrm{O}(1)$ and $f \in \mathrm{O}(\log\Delta)$.*

2.3.2 The Distributed Algorithm

In analogy to turning the greedy dominating set algorithm into a distributed algorithm in Section 2.2, the goal of this section is to transform the greedy ($\mathrm{LP_{DS}}$)-algorithm of Section 2.3.1 into a distributed algorithm for general covering and packing LPs. For our algorithm, we need the LPs (PP) and (DP) to be of the following special form:

$$\forall i,j \;:\; b_i = 1, \quad c_i \geq 1, \quad a_{ij} = 0 \text{ or } a_{ij} \geq 1. \tag{2.11}$$

The transformation to Condition (2.11) is done in two steps. First, every a_{ij} is replaced by $\hat{a}_{ij} := a_{ij}/b_i$ and b_i is replaced by 1. In the second step, the c_i and $\hat{a}_{ij}$ are divided by $\lambda_i := \min_j\{\hat{a}_{ji}, c_i\} \setminus \{0\}$. The optimal objective values of the transformed LPs are the same. A feasible solution for the transformed LP satisfying Condition (2.11) can be converted to a feasible solution of the original LP by dividing all x-values by the corresponding λ_i and by dividing the y-values by the corresponding b_i. This conserves the values of the objective functions. Note that the described transformation can be computed locally in a constant number of rounds. For the rest of this section, we assume that the coefficients of the LP are given according to Condition (2.11).

LP Approximation Algorithm for Primal Node v_i^p:

$x_i := 0$;
for $e_p := k_p - 2$ **to** $-\widehat{f}(v_i^p) - 1$ **by** -1 **do**
  for 1 **to** $\widehat{h}(v_i^p)$ **do**
    $(* \ \varphi_i := \frac{\widehat{C}(v_i^p)}{c_i} \sum_j a_{ji} r_j \ *)$
    for $e_d := k_d - 1$ **to** 0 **by** -1 **do**
      $\widetilde{\varphi}_i := \frac{\widehat{C}(v_i^p)}{c_i} \sum_j a_{ji} \widetilde{r}_j$;
      if $\widetilde{\varphi}_i \geq \widehat{\Phi}_p(v_i^p)^{e_p/k_p}$ **then**
        $x_i^+ := 1/\widehat{\Phi}_d(v_i^p)^{e_d/k_d}$; $x_i := x_i + x_i^+$;
      fi;
      send x_i^+, $\widetilde{\varphi}_i$ to dual neighbors;
      receive $\widetilde{r}_j$ from dual neighbors
    od;
    receive r_j from dual neighbors
  od
od;
$x_i := x_i / \min_{j \in N_i^p} \sum_\ell a_{j\ell} x_\ell$

LP Approximation Algorithm for Dual Node v_i^d:

$y_i := y_i^+ := w_i := f_i := 0$; $r_i := 1$;
for $e_p := k_p - 2$ **to** $-\widehat{f}(v_i^d) - 1$ **by** -1 **do**
  for 1 **to** $\widehat{h}(v_i^d)$ **do**
    $\widetilde{r}_i := r_i$;
    for $e_d := k_d - 1$ **to** 0 **by** -1 **do**
      receive x_j^+, $\widetilde{\varphi}_j$ from primal neighbors;
      $y_i^+ := y_i^+ + \widetilde{r}_i \sum_j a_{ij} x_j^+ / \widetilde{\varphi}_j$;
      $w_i^+ := \sum_j a_{ij} x_j^+$;
      $w_i := w_i + w_i^+$; $f_i := f_i + w_i^+$;
      if $w_i \geq 1$ **then** $\widetilde{r}_i := 0$ **fi;**
      send $\widetilde{r}_i$ to primal neighbors
    od;
    increase_duals();
    send r_i to primal neighbors
  od
od;
$y_i := y_i / \max_{j \in N_i^d} \frac{1}{c_j} \sum_\ell a_{\ell j} y_\ell$

Algorithm 3: Distributed LP Approximation Algorithm

procedure increase_duals():

1: **if** $w_i \geq 1$ **then**
2: **if** $f_i \geq \widehat{f}(v_i^d)$ **then**
3: $y_i := y_i + y_i^+$; $y_i^+ := 0$;
4: $r_i := 0$; $w_i := 0$
5: **else if** $w_i \geq 2$ **then**
6: $y_i := y_i + y_i^+$; $y_i^+ := 0$;
7: $r_i := r_i / \widehat{\Phi}_p(v_i^d)^{\lfloor w_i \rfloor / k_p}$
8: **else**
9: $\lambda := \max\{\widehat{\Phi}_d(v_i^d)^{1/k_d}, \widehat{\Phi}_p(v_i^d)^{1/k_p}\}$;
10: $y_i := y_i + \min\{y_i^+, r_i\lambda/\widehat{\Phi}_p(v_i^d)^{e_p/k_p}\}$;
11: $y_i^+ := y_i^+ - \min\{y_i^+, r_i\lambda/\widehat{\Phi}_p(v_i^d)^{e_p/k_p}\}$;
12: $r_i := r_i / \widehat{\Phi}_p(v_i^d)^{1/k_p}$
13: **fi**;
14: $w_i := w_i - \lfloor w_i \rfloor$
15: **fi**

Before describing the distributed algorithm for (PP) and (DP), we take another look at Algorithm 1 of Section 2.2.2. Algorithm 1 consists of two nested loops. In a single iteration ℓ of the outer loop, only nodes with degree at least $(\Delta+1)^{\ell/k}$ increase their x-value. Further, the total increase of each y-value obtained by dual fitting in a single iteration of the outer loop is at most $1/(\Delta+1)^{(\ell-1)/k}$. Up to a factor of $(\Delta+1)^{1/k}$, this is the y-value a node obtains if a single neighbor of degree $(\Delta+1)^{\ell/k}$ sets its x-value to 1 in the sequential greedy MDS algorithm. Hence, from the dual point of view, one iteration of the outer loop behaves like a single step of the greedy algorithm. Additionally, in each outer loop iteration of Algorithm 1, the maximum degree decreases by a factor of $(\Delta+1)^{1/k}$ (Lemma 2.2).

Algorithm 3 describes the distributed algorithm for (PP) and (DP). The structure of the innermost loop (e_d-loop) is similar to the structure of the inner loop of Algorithm 1. We will see that analogous to the above discussion, from a dual point of view, lines 4–18 of Algorithm 3 behave like a single step of the greedy ($\mathrm{LP}_{\mathrm{DS}}$)-algorithm described in the previous section. The code for increasing the dual values in line 17 of Algorithm 3 is given by the procedure **increase_duals()**. Because in the greedy (PP)-algorithm of Section 2.3.1, incrementing an x-value by 1 only reduces the requirements of the neighboring nodes by some factor, a single execution of lines 4–18 does not suffice to decrease the highest 'degree' φ_i by a large enough factor. We therefore execute lines 4–18 several times (second loop of the three nested loops) before decreasing e_p. Note that similar to ℓ in Algorithm 1, e_p characterizes the minimum φ_i a node must have to be active (cf. line 7).

value	primal node v_i^p	dual node v_i^d
Φ_p	$\max\left\{\rho^{k/5}, \widehat{\Phi}_p(v_i^p) \text{ by Algorithm 2}\right\}$	$\widehat{\Phi}_p(v_i^d) = \max_{v_j^p \in \Gamma(v_i^d)} \widehat{\Phi}_p(v_j^p)$
Φ_d	$\widehat{\Phi}_d(v_i^p) = \max_{v_j^d \in \Gamma(v_i^p)} \widehat{\Phi}_d(v_j^d)$	$\widehat{\Phi}_d(v_i^d)$ by Algorithm 2
f	$\widehat{f}(v_i^p) = \max_{v_j^d \in \Gamma v_i^p} \widehat{f}(v_j^d)$	$\left\lceil \frac{k_p+1}{\max\{\widehat{\Phi}_p(v_i^d)^{1/k_p}\}-1} \right\rceil$
h	$\widehat{h}(v_i^p) := \left\lceil 1 + \frac{k_p}{\widehat{\Phi}_p(v_i^p)^{1/k_p} \ln\left(\widehat{\Phi}_p(v_i^p)\right)} \right\rceil$	$\widehat{h}(v_i^d) = \max_{v_j^p \in \Gamma(v_i^d)} \widehat{h}(v_j^p)$
C	$\widehat{C}(v_i^p)$ by Algorithm 2	—

Table 2.1: Local estimates needed for Algorithm 3

Let us now take a closer look at the various variables occurring in Algorithm 3. Clearly, each primal node v_i^p needs a primal variable x_i. Each dual node v_i^d stores its dual variable y_i as well as the requirement r_i. Additionally, there are variables y_i^+, f_i, w_i, and $\widetilde{r}_i$. The variable y_i^+ keeps track of all dual increases which have not yet been added to y_i. Note that y_i is only updated in **increase_duals()**. During all k_d iterations of the innermost loop, only y_i^+ is increased. The variable f_i records the number of times the dual node v_i^d is covered, that is, f_i denotes the sum in the i^{th} inequality of the primal LP. The variable w_i represents the part of f_i for which the dual variable y_i has not yet been updated. In addition to r_i, the dual node v_i^d has a second requirement $\widetilde{r}_i$. In line 4, $\widetilde{r}_i$ is set to r_i. As soon as the the lefthand side of the i^{th} inequality of the primal LP is increased by at least 1, $\widetilde{r}_i$ is set to 0. As in the greedy fractional dominating set algorithm, φ_i denotes the sum of the requirements r_j of all dual neighbors v_j^d of a primal node v_i^p. Because we consider arbitrary covering and packing LPs, this sum is weighted by the coefficients a_{ji} of the linear program. The variable $\widetilde{\varphi}_i$ denotes the corresponding weighted sum for the requirements $\widetilde{r}_j$.

In addition to the mentioned variables, there are global quantities Φ_p, Φ_d, f, h, and C for which each node v stores estimates $\widehat{\Phi}_p(v)$, $\widehat{\Phi}_d(v)$, $\widehat{f}(v)$, $\widehat{h}(v)$, and $\widehat{C}(v)$, respectively. The values C, Φ_p, and Φ_d are defined as follows:

$$C := \max_i c_i, \quad \Phi_p := \max_i \frac{C}{c_i} \cdot \sum_{j=1}^{n} a_{ji}, \quad \text{and} \quad \Phi_d := \max_i \sum_{j=1}^{m} a_{ij}.$$

Thus C denotes the maximal weight of the primal objective function, The values of Φ_p and Φ_d correspond to the largest degree Δ in the dominating set case. Because we consider arbitrary covering and packing LPs, Φ_p and Φ_d are weighted sums of the coefficients of the LPs, that is, they are weighted versions of the maximum primal and dual degrees Δ_p and Δ_d of the network graph.

The values of f and h are defined as follows:

$$f := \left\lceil \frac{k_p + 1}{\Phi_p^{1/k_p} - 1} \right\rceil \quad \text{and} \quad h := \left\lceil 1 + \frac{k_p}{\Phi_p^{1/k_p} \ln \Phi_p} \right\rceil.$$

As in the fractional dominating set algorithm, f denotes the number of times each primal inequality has to be fulfilled (number of times each dual node has to be covered). The value h is the number of executions of lines 4–18 needed to decrease the maximal φ_i by a factor of Φ_p^{1/k_p} (see Lemma 2.13). The estimates for Φ_p, Φ_d, f, h, and C are computed as given by Table 2.1. Note that for the estimate $\widehat{\Phi}_p(v_i^p)$ we have introduced a new parameter $\rho > 1$. For those estimates which are computed by Algorithm 2, the nodes start the algorithm with the following initial estimates:

$$\widehat{C}(v_i^p) \;=\; c_i, \quad \widehat{\Phi}_p(v_i^p) \;=\; \frac{\widehat{C}(v_i^p)}{c_i} \sum_j a_{ji}, \quad \widehat{\Phi}_d(v_i^d) \;=\; \sum_j a_{ij}.$$

Note that the estimates for C have to be computed before estimating Φ_p. In the following, we assume that when computing the estimates for Φ_p, Φ_d, and C, the values for the parameter k in Algorithm 2 are k_p, k_d, and k_c, respectively. We denote the maximum ratio between the estimates of two primal or dual nodes at distance 2 by α_p, α_d, and α_c, respectively. By Theorem 2.6 and by the definition of the estimates in Table 2.1, we have $\alpha_p \leq \Phi_p^{2/k_p}$, $\alpha_d \leq \Phi_d^{2/k_d}$, and $\alpha_c \leq C^{2/k_c}$. Let us now look at the new parameter ρ. Because the outer two loops of Algorithm 3 are executed $k_p + \widehat{f}(v)$ and $\widehat{h}(v)$ times, the time complexity of the algorithm depends linearly on the largest estimates $f_{\max}$ and $h_{\max}$ for f and h, respectively. Without using ρ, for every k and Φ_p, $f_{\max}$ and $h_{\max}$ can become arbitrarily large because the ratio between the smallest estimate for Φ_p and Φ_p can get arbitrarily small in that case. For $h_{\max}$, note that for small ε $\ln(1+\varepsilon) \approx \varepsilon$. The parameter ρ gives an upper bound on the maximal possible estimates for f and h. We will see that the approximation ratio depends linearly on ρ, that is, besides bounding the time complexity from above, ρ also bounds the achievable approximation ratio from below. The maximal ratio between the estimates for f of adjacent nodes is given by the following lemma.

Lemma 2.12. *Let α_f be the maximal ratio between the estimates for f of two primal or dual nodes at distance* 2*. We have*

$$\alpha_f \;\leq\; \Phi_p^{2/k_p^2} + \frac{\Phi_p^{1/k_p} - 1}{k_p + 1} + \frac{\Phi_p^{2/k_p^2} - 1}{\rho^{1/5} - 1}.$$

Proof. It suffices to look at two dual nodes v_i^d and v_j^d at distance 2. Let $A_i := \max\{\widehat{\Phi}_p(v_i^d)^{1/k_p}\}$ and $A_j := \max\{\widehat{\Phi}_p(v_j^d)^{1/k_p}\}$. W.l.o.g., assume that $A_i \geq A_j$. We have

$$\frac{A_i}{A_j} \;\leq\; \frac{\widehat{\Phi}_p(v_i^d)^{1/k_p}}{\widehat{\Phi}_p(v_j^d)^{1/k_p}} \;\leq\; \alpha_p^{1/k_p} \;\leq\; \Phi_p^{2/k_p^2}.$$

The ratio between $\widehat{f}(v_j^d)$ and $\widehat{f}(v_i^d)$ now becomes

$$\begin{aligned}\frac{\widehat{f}(v_j^d)}{\widehat{f}(v_i^d)} &= \frac{\left\lceil \frac{k_p+1}{A_j-1} \right\rceil}{\left\lceil \frac{k_p+1}{A_i-1} \right\rceil} \leq \frac{1+\frac{k_p+1}{A_j-1}}{\frac{k_p+1}{A_i-1}} \leq \frac{A_i-1}{k_p+1} + \frac{\Phi_p^{1/k_p^2} A_j - 1}{A_j - 1} \\ &= \frac{A_i-1}{k_p+1} + \Phi_p^{2/k_p^2} + \frac{\Phi_p^{2/k_p^2}-1}{A_j-1} \leq \frac{\Phi_p^{1/k_p}-1}{k_p+1} + \Phi_p^{2/k_p^2} + \frac{\Phi_p^{2/k_p^2}-1}{\rho^{1/5}-1}.\end{aligned}$$

□

Before starting with the analysis of Algorithm 3, we need to make a remark concerning the implementation of the algorithm. The algorithm is written in a way which optimizes its readability. This causes imprecision in some formulations. In particular, the number of iterations of the outer two for-loops depends on the estimates of f and h, respectively. Hence, the number of iterations varies for different nodes. However, we want neighboring nodes to simultaneously go to the next outer loop iteration, that is, neighboring nodes should decrement k_p at the same time. We therefore have to locally synchronize the iterations of different nodes. A node can start a new iteration of a for loop as soon as all neighbors have finished their current iteration. To enhance readability, the necessary control messages are left out in the formulation of Algorithm 3.

2.3.3 Analysis

After giving an overview, we now come to the analysis of Algorithm 3. In the following, we prove a series of lemmas which finally lead to the desired results about the approximation ratio and the time complexity of Algorithm 3. Analogously to Lemma 2.2 in Section 2.2.2, the first lemma gives an upper bound on the 'degree' φ_i of a primal node v_i^p.

Lemma 2.13. *For each primal node v_i^p, at all times during Algorithm 3,*

$$\varphi_i \leq \widehat{\Phi}_p(v_i^p)^{(e_p+2)/k_p}.$$

Proof. We prove the lemma by induction over the iterations of the outermost loop (e_p-loop). For $e_p = k_p - 2$, the lemma follows from the definitions of φ_i, $\widehat{\Phi}_p(v_i^p)$, and r_j.

To see how fast φ_i decreases, we have to look at the behavior of the innermost loop (e_d-loop). The value of φ_i is $\widehat{C}(v_i^p)/c_i$ times the weighted sum $a_{ji}r_j$ of the requirements of all dual neighbors v_j^d of v_i^p. The variable $\widetilde{\varphi}_i$ is the same sum but only for the dual neighbors for which the corresponding primal inequality has not been fulfilled since the last time **increase_duals()** was called ($w_i < 1$ and $\widetilde{r}_i > 0$). When **increase_duals()** is called (i.e., after the last iteration of the e_d-loop), it holds that $\widetilde{\varphi}_i < \widehat{\Phi}_p(v_i^p)^{e_p/k_p}$. If not, x_i^+ would have been set to 1 in the last e_d-loop iteration ($e_d = 0$). Because all $a_{ij} \geq 1$ and all $b_j = 1$, all dual

neighbors v_j^d of v_i^p would then have $w_j \geq 1$ and therefore $\widetilde{r}_j = 0$ after the last e_d-loop iteration. Thus, if $\widetilde{\varphi}_i \geq \widehat{\Phi}_p(v_i^p)^{e_p/k_p}$ before the last e_d-loop iteration, then $\widetilde{\varphi}_i = 0$ when **increase_duals()** is called.

All dual nodes v_j^d which set $\widetilde{r}_j := 0$ divide r_j by at least $\widehat{\Phi}_p(v_j^d)^{1/k_p} \geq \widehat{\Phi}_p(v_i^p)^{1/k_p}$ while executing **increase_duals()**. Therefore, after the call to **increase_duals()**, we have

$$\varphi_i' \leq \widetilde{\varphi}_i + \frac{\varphi_i - \widetilde{\varphi}_i}{\widehat{\Phi}_p(v_i^p)^{1/k_p}} \leq \widehat{\Phi}_p(v_i^p)^{e_p/k_p} + \frac{\varphi_i - \widehat{\Phi}_p(v_i^p)^{e_p/k_p}}{\widehat{\Phi}_p(v_i^p)^{1/k_p}},$$

where φ_i and φ_i' denote the values before and after executing **increase_duals()**, respectively. Before going to the next e_p-loop iteration, the lines 4–18 are executed $\widehat{h}(v_i^p)$ times. By the induction hypothesis, $\varphi_i \leq \widehat{\Phi}_p(v_i^p)^{(e_p+2)/k_p}$ before the $\widehat{h}(v_i^p)$ iterations of the inner part and therefore, after the $\widehat{h}(v_i^p)$ iterations, we have

$$\varphi_i \;\leq\; \widehat{\Phi}_p(v_i^p)^{e_p/k_p} + \frac{\widehat{\Phi}_p(v_i^p)^{(e_p+2)/k_p} - \widehat{\Phi}_p(v_i^p)^{e_p/k_p}}{\widehat{\Phi}_p(v_i^p)^{\widehat{h}(v_i^p)/k_p}}. \tag{2.12}$$

To prove the lemma, we have to show that the right-hand side of Inequality (2.12) is at most $\widehat{\Phi}_p(v_i^p)^{(e_p+1)/k_p}$. Dividing the right-hand side of Inequality (2.12) and $\widehat{\Phi}_p(v_i^p)^{(e_p+1)/k_p}$ by $\widehat{\Phi}_p(v_i^p)^{e_p/k_p}$, this results in

$$\left(\widehat{\Phi}_p(v_i^p)^{1/k_p} - 1\right) \cdot \left(\widehat{\Phi}_p(v_i^p)^{1/k_p} + 1\right) \leq \left(\widehat{\Phi}_p(v_i^p)^{1/k_p} - 1\right) \cdot \widehat{\Phi}_p(v_i^p)^{\widehat{h}(v_i^p)/k_p}$$

which is true for

$$\widehat{h}(v_i^p) \;\geq\; \frac{\ln\left(\widehat{\Phi}_p(v_i^p)^{1/k_p} + 1\right)}{\ln\left(\widehat{\Phi}_p(v_i^p)^{1/k_p}\right)}.$$

Because the logarithm is a concave function, we have $\ln(x+1) \leq \ln(x) + 1/x$ and therefore

$$\begin{aligned}\frac{\ln\left(\widehat{\Phi}_p(v_i^p)^{1/k_p} + 1\right)}{\ln\left(\widehat{\Phi}_p(v_i^p)^{1/k_p}\right)} \;&\leq\; 1 + \frac{1}{\widehat{\Phi}_p(v_i^p)^{1/k_p} \ln\left(\widehat{\Phi}_p(v_i^p)^{1/k_p}\right)} \\ &=\; 1 + \frac{k_p}{\widehat{\Phi}_p(v_i^p)^{1/k_p} \ln\left(\widehat{\Phi}_p(v_i^p)\right)} \;\leq\; \widehat{h}(v_i^p)\end{aligned}$$

by the definition of $\widehat{h}(v_i^p)$. □

Lemma 2.14. *After line 12 of Algorithm 3, for each dual node v_i^d, $\widetilde{r}_i = 0$ or*

$$w_i^+ := \sum_j a_{ij} x_j^+ \leq \alpha_d \widehat{\Phi}_d(v_i^d)^{1/k_d}.$$

Proof. For the sake of contradiction, assume that $\widetilde{r}_i > 0$ and $w_i^+ > \alpha_d \widehat{\Phi}_d(v_i^d)^{1/k_d}$. Let $w_i'^+$ be the sum of the $a_{ij}x_j^+$ of the preceding iteration of the e_d-loop. All primal variables which are increased in line 8 have also been increased in the preceding iteration because the $\widetilde{\varphi}_j$ of the primal neighbors of v_i^d can only have decreased. Therefore

$$w_i'^+ \geq \sum_j \frac{a_{ij}}{\widehat{\Phi}_d(v_j^p)^{1/k_d}} \geq \frac{w_i^+}{\alpha_d \widehat{\Phi}_d(v_i^d)^{1/k_d}} > 1$$

which is a contradiction to $\widetilde{r}_i > 0$ because $w_i'^+ > 1$ implies that $\widetilde{r}_i$ has been set to 0 in the preceding iteration of the e_d-loop. For the first iteration of the e_d-loop ($e_d = k_d - 1$), we have

$$w_i^+ = \sum_j a_{ij}x_j^+ = \sum_j \frac{a_{ij}}{\widehat{\Phi}_d(v_j^p)^{(k_d-1)/k_d}} \leq \frac{\sum_j a_{ij}}{\widehat{\Phi}_d(v_i^d)^{(k_d-1)/k_d}} \leq \frac{\widehat{\Phi}_d(v_i^d)}{\widehat{\Phi}_d(v_i^d)^{(k_d-1)/k_d}},$$

which completes the proof. Note that by definition (Table 2.1), $\widehat{\Phi}_d(v_j^p) \geq \widehat{\Phi}_d(v_i^d)$ if $a_{ij} \neq 0$. □

Lemma 2.15. *Each time a dual v_i^d node enters procedure **increase_duals()**, the value y_i^+ can be bounded as follows:*

$$y_i^+ \leq \alpha_p r_i \cdot \frac{w_i}{\widehat{\Phi}_p(v_i^d)^{e_p/k_p}} \text{ and } y_i^+ \leq \alpha_p r_i \cdot \frac{\alpha_d \widehat{\Phi}_d(v_i^d)^{1/k_d} + 1}{\widehat{\Phi}_p(v_i^d)^{e_p/k_p}}. \tag{2.13}$$

Proof. We start by showing the left inequality. First, we prove that the condition can only be violated inside **increase_duals()**. Outside, r_i is not changed. Further y_i^+ and w_i are always increased simultaneously in lines 11 and 13. Because $\widetilde{r}_i \leq r_i$ and $\widetilde{\varphi}_j \geq (\alpha_p \widehat{\Phi}_p(v_i^d))^{e_p/k_p}$, these increases do not violate Condition (2.13). Decreasing e_p only increases the right-hand side of the inequality and therefore Condition (2.13) is not invalidated by this either.

Inside **increase_duals()**, we consider the 3 cases $w_i < 1$, $1 \leq w_i < 2$, and $w_i \geq 2$. If $w_i < 1$, nothing happens and we are done. If $w_i \geq 2$ or $f_i \geq f$, y_i^+ is set to 0 and hence Condition (2.13) trivially holds. The interesting case is when $1 \leq w_i < 2$.

We assume that Condition (2.13) holds before entering **increase_duals()**. We define $w_i' := w_i - 1$ to be the fractional part of w_i. Inside **increase_duals()** $r_i\lambda/\widehat{\Phi}_p(v_i^d)^{e_p/k_p}$ is subtracted from y_i^+ and r_i is divided by $\widehat{\Phi}_p(v_i^d)^{1/k_p}$. Condition (2.13) is true after **increase_duals()** if the following inequality holds (r_i before the division):

$$y_i^+ - \frac{r_i\lambda}{\widehat{\Phi}_p(v_i^d)^{e_p/k_p}} \leq \alpha_p r_i \frac{1 + w_i' - \lambda}{\widehat{\Phi}_p(v_i^d)^{e_p/k_p}} \leq \frac{\alpha_p r_i}{\widehat{\Phi}_p(v_i^d)^{1/k_p}} \frac{w_i'}{\widehat{\Phi}_p(v_i^d)^{e_p/k_p}}.$$

This gives

$$\begin{aligned}\lambda\widehat{\Phi}_p(v_i^d)^{1/k_p} - \widehat{\Phi}_p(v_i^d)^{1/k_p} - w_i'\widehat{\Phi}_p(v_i^d)^{1/k_p} + w_i' &\geq \\ \widehat{\Phi}_p(v_i^d)^{2/k_p} - \widehat{\Phi}_p(v_i^d)^{1/k_p} - w_i'\widehat{\Phi}_p(v_i^d)^{1/k_p} + w_i' &= \\ \left(\widehat{\Phi}_p(v_i^d)^{1/k_p} - 1\right)\cdot\left(\widehat{\Phi}_p(v_i^d)^{1/k_p} - w_i'\right) &\geq 0,\end{aligned}$$

which is true because $\lambda \geq \widehat{\Phi}_p(v_i^d)^{1/k_p} \geq 1$ and $w_i' < 1$.

To prove the right inequality of Condition (2.13), note that y_i^+ is only increased in the e_d-loop as long as $w_i < 1$ before entering the loop. Otherwise, $\widetilde{r}_i$ would have been set to zero. The second inequality of the lemma now follows from the first inequality and from Lemma 2.14. □

Lemma 2.16. *Let v_i^p be a primal node and let $Y_i := \sum_j a_{ji}y_j$ be the weighted sum of the y-values of its dual neighbors. Further, let Y_i^+ be the increase of Y_i and φ_i^- be the decrease of φ_i during an execution of **increase_duals()**. We have*

$$Y_i^+ \leq \frac{\alpha_p\widehat{\Phi}_p(v_i^p)^{3/k_p}\cdot\max\{\widehat{\Phi}_p(v_i^p)^{1/k_p},\widehat{\Phi}_d(v_i^p)^{1/k_d}\}}{\varphi_i(\widehat{\Phi}_p(v_i^p)^{1/k_p}-1)}\cdot\frac{c_i}{\widehat{C}(v_i^p)}\cdot\varphi_i^-.$$

Proof. We prove the lemma by showing that the inequality holds for every dual neighbor v_j^d of v_i^p. Let β_j be the increase of y_j and let r_j^- be the decrease of r_j. We show that

$$\beta_j \leq \frac{\alpha_p\widehat{\Phi}_p(v_i^d)^{1/k_p}\cdot\max\{\widehat{\Phi}_p(v_i^p)^{1/k_p},\widehat{\Phi}_d(v_i^p)^{1/k_d}\}}{\widehat{\Phi}_p(v_i^p)^{e_p/k_p}(\widehat{\Phi}_p(v_i^p)^{1/k_p}-1)}\cdot r_j^-. \tag{2.14}$$

The lemma then follows because $\varphi_i \leq \widehat{\Phi}_p(v_i^p)^{(e_p+2)/k_p}$ (Lemma 2.13) and because

$$Y_i^+ = \sum_j a_{ji}\beta_j \quad\text{and}\quad \varphi_i^- = \frac{\widehat{C}(v_i^p)}{c_i}\sum_j a_{ji}r_j^-.$$

To prove Inequality (2.14), we again consider the cases where $w_j \geq 2$ and where $1 \leq w_j < 2$. If $w_j \geq 2$, by Lemma 2.15, $\beta_j = y_j^+ \leq \alpha_p r_j(1+\alpha_d\widehat{\Phi}_d(v_i^p)^{1/k_d})/\widehat{\Phi}_p(v_i^p)^{e_p/k_p}$. The requirement r_j is divided by at least $\widehat{\Phi}_p(v_i^p)^{2/k_p}$ and therefore $r_j^- \geq r_j(\widehat{\Phi}_p(v_i^p)^{2/k_p}-1)/\widehat{\Phi}_p(v_i^p)^{2/k_p}$. Together, we get

$$\begin{aligned}\beta_j &\leq \alpha_p\alpha_d\cdot\frac{1+\widehat{\Phi}_d(v_i^p)^{1/k_d}}{\widehat{\Phi}_p(v_i^p)^{e_p/k_p}}\cdot\frac{\widehat{\Phi}_p(v_i^p)^{2/k_p}}{\widehat{\Phi}_p(v_i^p)^{2/k_p}-1}\cdot r_j^- \\ &\leq \frac{\alpha_p\alpha_d\left(1+\widehat{\Phi}_p(v_i^p)^{1/k_p}\right)\widehat{\Phi}_p(v_i^p)^{1/k_p}\max\{\widehat{\Phi}_p(v_i^p)^{1/k_p},\widehat{\Phi}_d(v_i^p)^{1/k_d}\}}{\widehat{\Phi}_p(v_i^p)^{e_p/k_p}\left(\widehat{\Phi}_p(v_i^p)^{1/k_p}+1\right)\left(\widehat{\Phi}_p(v_i^p)^{1/k_p}-1\right)}r_j^-.\end{aligned}$$

Note that for all $A, B \geq 0$, $(1+A)B \leq (1+B)\max\{A,B\}$. For $1 \leq w_j < 2$, the proof is along the same lines. Here,

$$\beta_j \;\leq\; \alpha_p\alpha_d r_j \max\{\widehat{\Phi}_p(v_i^p)^{1/k_p}, \widehat{\Phi}_d(v_i^p)^{1/k_d}\}/\widehat{\Phi}_p(v_i^p)^{e_p/k_p}$$

and $r_j^- = r_j(\widehat{\Phi}_p(v_i^p)^{1/k_p} - 1)/\widehat{\Phi}_p(v_i^p)^{1/k_p}$. Again, we obtain Inequality (2.14):

$$\beta_j \leq \alpha_p\alpha_d \frac{\max\{\widehat{\Phi}_p(v_i^p)^{1/k_p}, \widehat{\Phi}_d(v_i^p)^{1/k_d}\}}{\widehat{\Phi}_p(v_i^p)^{e_p/k_p}} \cdot \frac{\widehat{\Phi}_p(v_i^p)^{1/k_p}}{\widehat{\Phi}_p(v_i^p)^{1/k_p} - 1} \cdot r_j^-.$$

We do not have to consider the case $f_j \geq \widehat{f}(v_j^d)$ explicitly because the same analysis as for $w_j \geq 2$ applies in this case. □

Lemma 2.17. *Let v_i^p be a primal node and let $Y_i = \sum_j a_{ji}y_j$ be the weighted sum of the y-values of the dual neighbors of v_i^p. After the main part of the algorithm (i.e., after the loops at line 20),*

$$Y_i \leq \alpha_p\alpha_d \frac{c_i}{\widehat{C}(v_i^p)}(k_p + \widehat{f}(v_i^p) + 1)\widehat{\Phi}_p(v_i^p)^{3/k_p} \max\left\{\widehat{\Phi}_p(v_i^p)^{1/k_p}, \widehat{\Phi}_d(v_i^p)^{1/k_d}\right\}.$$

Proof. For simplicity, we define

$$Q := \frac{\alpha_p\alpha_d}{\widehat{C}(v_i^p)}\widehat{\Phi}_p(v_i^p)^{3/k_p} \max\{\widehat{\Phi}_p(v_i^p)^{1/k_p}, \widehat{\Phi}_d(v_i^p)^{1/k_d}\}.$$

Before φ_i is decreased for the last time, we have $\varphi_i \geq 1/\widehat{\Phi}_p(v_i^p)^{(\widehat{f}(v_i^p)-1)/k_p}$ because at least one r_j in the dual neighborhood of v_i^p has to be greater than 0. If we assume that the last time φ_i is decreased, it is only reduced to $\varphi_i = 1/\widehat{\Phi}_p(v_i^p)^{(\widehat{f}(v_i^p)+1)/k_p}$, Lemma 2.16 still holds. The analysis is exactly the same as for the case $w_j \geq 2$ in Lemma 2.16. Using the same technique as for bounding Y_i in the analysis of the sequential greedy fractional dominating set algorithm of Section 2.3.1, we can also bound Y_i here. By Lemma 2.16, decreasing φ_i by φ_i^- enlarges Y_i by $Y_i^+ \leq c_iQ/(\widehat{\Phi}_p(v_i^p)^{1/k_p} - 1) \cdot \varphi_i^-/\varphi_i$. The given upper bound on Y_i^+ can be represented by the area of a rectangle as in Figure 2.2. Analogously to the analysis of the sequential algorithm, we can upper-bound Y_i by the following integral.

$$\begin{aligned}
Y_i &\leq \frac{c_iQ}{\widehat{\Phi}_p(v_i^p)^{1/k_p} - 1} \cdot \int_{\frac{1}{\widehat{\Phi}_p(v_i^p)^{(\widehat{f}(v_i^p)+1)/k_p}}}^{\widehat{\Phi}_p(v_i^p)} \frac{1}{x}dx \\
&= \frac{c_iQ\ln\left(\widehat{\Phi}_p(v_i^p)^{(k_p+\widehat{f}(v_i^p)+1)/k_p}\right)}{\widehat{\Phi}_p(v_i^p)^{1/k_p} - 1} \\
&= \frac{c_i(k_p + \widehat{f}(v_i^p) + 1)Q\ln\left(\widehat{\Phi}_p(v_i^p)^{1/k_p}\right)}{\widehat{\Phi}_p(v_i^p)^{1/k_p} - 1} \\
&\leq c_i(k_p + \widehat{f}(v_i^p) + 1)Q.
\end{aligned}$$

The last inequality follows from $\ln(1+t) \leq t$. □

Lemma 2.18. *At the end of Algorithm 3, we have for each dual node* v_i^d

$$r_i = 0 \text{ and } f_i \geq \widehat{f}(v_i^d).$$

Proof. When entering the e_p-loop for the last time, by Lemma 2.13,

$$\widehat{\Phi}_p(v_j^p)^{(-\widehat{f}(v_j^p)+1)/k_p} \geq \varphi_j \geq \sum_i a_{ij} r_i \geq \sum_{i \in N_j^p} r_i$$

for every primal node v_j^p. The value of φ_j can only be greater than 0 if there is exactly one r_i in the dual neighborhood of v_j^p which is greater than zero. If r_i is still greater than 0 when $e_d = 0$, x_j will be increased by 1, which makes $w_j \geq 1$ and therefore $r_i = 0$ after the next call to **increase_duals()**.

f_i counts the number of times the i^{th} constraint of (PP) is satisfied. It is increased together with w_i in line 13 of Algorithm 3. Every time the integer part of w_i is increased, r_i is divided by $\widehat{\Phi}_p(v_i^d)^{\lfloor w_i \rfloor / k_p}$ and w_i is set to $w_i - \lfloor w_i \rfloor$. Therefore, $r_i = 0$ implies $f_i \geq \widehat{f}(v_i^d)$. □

To compare the objective functions of (PP) and (DP), we define $y_j^{(i)}$ to be the part of y_j which is contributed by x_i where v_j^d and v_i^p are neighbors in G. Hence, each time x_i is increased by x_i^+, $y_j^{(j)}$ is increased by $a_{ji}\widetilde{r}_j x_i^+ / \widetilde{\varphi}_i$.

Lemma 2.19. *After the main part of Algorithm 3 (i.e., after the loops at line 20), we have*

$$\sum_{i | a_{ji} \neq 0} y_j^{(i)} = y_j \text{ and } c_i x_i = \widehat{C}(v_i^p) \sum_{j | a_{ji} \neq 0} y_j^{(i)}.$$

Proof. We first prove the left equation. By Lemma 2.18, all y_j^+ are 0 in the end and thus y_j is equal to the sum of all increases of y_j^+. It therefore suffices to show that each time y_j^+ is increased in line 11, $\sum_i y_j^{(i)}$ is increased by the same amount. However, it follows from the above definition of $y_j^{(i)}$ and by line 11 of Algorithm 3 that y_j and $\sum_i y_j^{(i)}$ are both increased by $\widetilde{r}_j \sum_i a_{ji} x_i / \widetilde{\varphi}_i$.

For the right equation, let v_i^p be a primal node which increases x_i by x_i^+ (line 8). All dual neighbors v_j^d of v_i^p increase $y_j^{(i)}$ by $y_j^{(i)+} := a_{ji}\widetilde{r}_j x_i^+ / \widetilde{\varphi}_i$. Hence,

$$\sum_{j | a_{ji} \neq 0} y_j^{(i)+} = \frac{x_i^+}{\widetilde{\varphi}_i} \sum_j a_{ji}\widetilde{r}_j = x_i^+ \frac{\sum_j a_{ji}\widetilde{r}_j}{\frac{\widehat{C}(v_i^p)}{c_i} \sum_j a_{ji}\widetilde{r}_j} = \frac{c_i}{\widehat{C}(v_i^p)} x_i^+.$$

The lemma follows by summing over all increases x_i^+ of x_i. □

Lemma 2.20. *Algorithm 3 approximates* (PP) *and* (DP) *by a factor of*

$$\alpha_p \alpha_d \alpha_c \alpha_f^2 \rho \Phi_p^{4/k_p} \max\left\{\Phi_p^{1/k_p}, \Phi_d^{1/k_d}\right\}$$

Proof. In line 21 of Algorithm 3, each primal node v_i^p divides x_i by the largest possible number such that (PP) remains feasible and each dual node v_j^d divides y_j by the smallest necessary number such that (DP) becomes feasible. Hence, the solutions computed by Algorithm 3 are feasible. To obtain the approximation ratio of Algorithm 3, we have a closer look at the number by which the x-values and the y-values are divided in line 21.

Let x_i' and y_j' be the values of x_i and y_j after line 21, respectively. The corresponding values before line 21 are denoted by x_i and y_j, respectively. By Lemma 2.17

$$\begin{aligned}\frac{y_j}{y_j'} &\leq \alpha_p\alpha_d \max_{i|a_{ji}\neq 0}\left(\frac{(k_p+\widehat{f}(v_i^p)+1)\widehat{\Phi}_p(v_i^p)^{3/k_p}\max\left\{\widehat{\Phi}_p(v_i^p)^{1/k_p},\widehat{\Phi}_d(v_i^p)^{1/k_d}\right\}}{\widehat{C}(v_i^p)}\right)\\ &\leq \alpha_p\rho^{3/5}\Phi_p^{3/k_p}\max\left\{\rho^{1/5}\Phi_p^{1/k_p},\Phi_d^{1/k_d}\right\}\cdot\max_{i|a_{ji}\neq 0}\left(\frac{k_p+\widehat{f}(v_i^p)+1}{\widehat{C}(v_i^p)}\right).\end{aligned}$$

For simplicity, we define

$$\gamma := \alpha_p\alpha_d\rho^{3/5}\Phi_p^{3/k_p}\max\{\rho^{1/5}\Phi_p^{1/k_p},\Phi_d^{1/k_d}\}.$$

Each time an inequality of (PP) is satisfied during the execution of Algorithm 3, the value f_j of the corresponding dual node v_j^d is incremented by 1. In the end, $f_j \geq \widehat{f}(v_j^d)$, that is, the j^{th} inequality of (PP) is satisfied at least $\widehat{f}(v_j^d)$ times. Hence, we have

$$\frac{x_i}{x_i'} \geq \min_{j|a_{ji}\neq 0}\widehat{f}(v_j^d) \geq \frac{\widehat{f}(v_i^p)}{\alpha_f}.$$

Applying Lemma 2.19 and the obtained bounds on the ratios y_j/y_j' and x_i/x_i', we thus obtain

$$\begin{aligned}\sum_{i=1}^m c_i x_i' &\leq \alpha_f\sum_{i=1}^m\frac{c_i x_i}{\widehat{f}(v_i^p)} = \alpha_f\sum_{i=1}^m\frac{\widehat{C}(v_i^p)}{\widehat{f}(v_i^p)}\sum_{j|a_{ji}\neq 0}y_j^{(i)}\\ &\leq \alpha_f\gamma\sum_{i=1}^m\frac{\widehat{C}(v_i^p)}{\widehat{f}(v_i^p)}\sum_{j|a_{ji}\neq 0}\max_{i|a_{ji}\neq 0}\left(\frac{k_p+\widehat{f}(v_i^p)+1}{\widehat{C}(v_i^p)}\right)y_j'^{(i)}\\ &\leq \alpha_f^2\alpha_c\gamma\sum_{i=1}^m\frac{k_p+\widehat{f}(v_i^p)+1}{\widehat{f}(v_i^p)}\sum_{j|a_{ji}\neq 0}y_j'^{(i)}\\ &\leq \alpha_f^2\alpha_c\gamma\sum_{i=1}^m\frac{k_p+\frac{k_p+1}{\widehat{\Phi}_p(v_i^p)^{1/k_p}-1}+1}{\frac{k_p+1}{\widehat{\Phi}_p(v_i^p)^{1/k_p}-1}}\sum_{j|a_{ji}\neq 0}y_j'^{(i)}\\ &= \alpha_f^2\alpha_c\gamma\sum_{i=1}^m\widehat{\Phi}_p(v_i^p)^{1/k_p}\sum_{j|a_{ji}\neq 0}y_j'^{(i)} \leq \alpha_f^2\alpha_c\gamma\rho^{1/5}\Phi_p^{1/k_p}\sum_{j=1}^n y_j'.\end{aligned}$$

Thereby, $y_j'^{(i)}$ is defined as the contribution of x_i' to y_i', that is, $y_j'^{(i)} = y_j^{(i)} \cdot y_j'/y_j$. Plugging in the definition of γ, we get

$$\sum_{i=1}^{m} c_i x_i' \;\leq\; \alpha_p \alpha_d \alpha_c \alpha_f^2 \rho^{4/5} \Phi_p^{4/k_p} \max\left\{\rho^{1/5} \Phi_p^{1/k_p}, \Phi_d^{1/k_d}\right\} \sum_{j=1}^{n} y_j',$$

which concludes the proof. □

Theorem 2.21. *For arbitrary $k_p, k_d, k_c \geq 1$ and $\rho > 1$, Algorithm 3 approximates* (PP) *and* (DP) *by a factor of*

$$\rho \cdot C^{1/k_c} \cdot \Phi_p^{\frac{\mathrm{O}(1)}{k_p}} \cdot \Phi_d^{\frac{1}{k_d}} \cdot \left(1 + \frac{\Phi_p^{1/k_p^2} - 1}{\rho^{1/5} - 1}\right).$$

The time complexity of Algorithm 3 is

$$\mathrm{O}\left(k_c + k_p k_d \left(1 + \frac{1}{\rho^{1/5} - 1}\right)\left(1 + \frac{1}{\rho^{1/5} \log \rho}\right)\right).$$

For constant ρ, this simplifies to $\mathrm{O}(k_c + k_p k_d)$.

Proof. To obtain the time complexity of Algorithm 3, note that the number of rounds is proportional to the number of iterations of the innermost loop (e_d-loop). Each iteration of the innermost loop takes two rounds. Hence, the algorithm has time complexity $\mathrm{O}(k_d(k_p + f_{\max})h_{\max})$, where $f_{\max}$ and $h_{\max}$ denote the largest estimates for f and h, respectively. By the definition of the estimates for Φ_p, f, and h, we have

$$f_{\max} \;\leq\; \left\lceil \frac{k_p + 1}{\rho^{1/5} - 1} \right\rceil \quad \text{and} \quad h_{\max} \;\leq\; \left\lceil 1 + \frac{k_p}{\rho^{1/5} \ln\left(\rho^{k_p/5}\right)} \right\rceil.$$

The approximation ratio directly follows from Lemmas 2.12 and 2.20. □

Corollary 2.22. *For sufficiently small ε (ε at most constant), Algorithm 3 computes a $(1+\varepsilon)$-approximation for* (PP) *and* (DP) *in $\mathrm{O}(\log C/\varepsilon + \log \Phi_p \log \Phi_d/\varepsilon^4)$ rounds. In particular, a constant factor approximation can be achieved in time $\mathrm{O}(\log C + \log \Phi_p \log \Phi_d)$.*

Proof. We choose k_c, k_p, k_d, and ρ such that $C^{1/k_c} = \Phi_p^{1/k_p} = \Phi_d^{1/k_d} = \rho = 1 + \varepsilon'$. By this, we get

$$k_c = \frac{\log C}{\log(1+\varepsilon')}, \quad k_p = \frac{\log \Phi_p}{\log(1+\varepsilon')}, \quad \text{and} \quad k_d = \frac{\log \Phi_d}{\log(1+\varepsilon')}.$$

For constant ε', we then have $k_c = \mathrm{O}(\log C)$, $k_p = \mathrm{O}(\log \Phi_p)$, and $k_d = \mathrm{O}(\log \Phi_d)$. For ε' at most constant, first order Taylor approximation gives $\ln(1+\varepsilon') \in \Theta(\varepsilon')$. We can therefore bound the expressions for k_c, k_p, and k_d

by $k_c \in \mathrm{O}(\log C/\varepsilon')$, $k_p \in \mathrm{O}(\log \Phi_p/\varepsilon')$, and $k_d \in \mathrm{O}\log \Phi_d/\varepsilon')$. Let us now determine the approximation ratio achieved by this choice of the parameters. We first look at the term Φ_p^{1/k_p^2}, which occurs in the last factor of the approximation ratio of Theorem 2.21. There is a constant c such that

$$\Phi_p^{1/k_p^2} \;\leq\; \Phi_p^{\frac{c\varepsilon'^2}{\ln^2(\Phi_p)}} \;=\; e^{\frac{c\varepsilon'^2}{\ln \Phi_p}} \;\leq\; e^{c\varepsilon'^2} \;\in\; 1+\mathrm{O}(\varepsilon'^2).$$

The approximation ratio therefore is

$$\rho \cdot C^{1/k_c} \cdot \Phi_p^{\frac{\mathrm{O}(1)}{k_p}} \cdot \Phi_d^{\frac{1}{k_d}} \cdot \left(1+\frac{\Phi_p^{1/k_p^2}-1}{\rho^{1/5}-1}\right) \;\in\; 1+\mathrm{O}(\varepsilon').$$

We can therefore choose $\varepsilon \in \mathrm{O}(\varepsilon')$. The given time complexity is obtained because

$$\frac{1}{\rho^{1/5}-1} \;\in\; \mathrm{O}\left(\frac{1}{\varepsilon'}\right) \quad \text{and} \quad \frac{1}{\rho^{1/5}\ln\rho} \;\in\; \mathrm{O}\left(\frac{1}{\varepsilon'}\right).$$

This concludes the proof. □

Message Size: Algorithm 3 can be implemented such that all messages are of size $\mathrm{O}(k_c+k_p+k_d+\log m+\log n+\log\gamma)$. Thus, as long as γ is at most polynomial in m and n, messages are of size $\mathrm{O}(\log n + 1/\varepsilon)$. If the algorithm is slightly adapted, it is even possible to bound all messages by $\mathrm{O}(\log n)$, independently of γ and ε. Further, note that no node ever has to send two different messages at the same time. In each round, all nodes can send the same message to all neighbors. Therefore, Algorithm 3 really works in the $\mathcal{CONGEST}_{\mathrm{BC}}$ model as claimed at the beginning of Chapter 2.

Dependence on Coefficients: Let $a_{\max} := \max_{i,j}\{a_{ij}, b_i, c_i\}$ be the largest coefficient of (PP) and (DP). Analogously, $a_{\min} := \min_{i,j}(\{a_{ij}, b_i, c_i\} \setminus \{0\})$ is the smallest non-zero coefficient of (PP) and (DP). For $\gamma := a_{\max}/a_{\min}$, we have $C \leq \gamma$, $\Phi_p \leq \gamma\Delta_p$, and $\Phi_d \leq \gamma\Delta_d$. The time complexity of the above corollary can hence be bounded by $\mathrm{O}((\log\gamma + \log\Delta_p \log\Delta_d)/\varepsilon^4)$. Consequently, running time and approximation ratio are dependent on the values of the coefficients. It is possible to get rid of this dependence by using methods which are described in [17, 97].

Comparison to the Algorithm of [17]: Similarly to Algorithm 3, in [17] a local approximation scheme for covering and packing LPs for the $\mathcal{CONGEST}_{\mathrm{BC}}$ model has been presented. Besides using different techniques, Algorithm 3 differs from the algorithm of [17] in the following points. The algorithm of [17] computes a $(1+\varepsilon)$-approximation in time $\mathrm{O}(\log^2(\gamma m)\log(\gamma mn/\varepsilon)/\varepsilon^3)$, where γ is defined as above. Hence, for constant ε our algorithm is faster by a factor

of $\Theta(\log m)$. Further, Algorithm 3 allows to establish a trade-off between time complexity T and approximation quality for all T, whereas the algorithm of [17] has time complexity $\Omega(\log^2(\gamma m))$ independent of the achieved approximation ratio.

2.4 A Fast Algorithm Based on Network Decomposition

In [93], Linial and Saks presented a randomized distributed algorithm for a weak-diameter network decomposition. We use their algorithm to decompose the linear program into sub-programs which can be solved locally in the $\mathcal{LOCAL}$ model. For a general graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with n nodes and for each color of the decomposition, the algorithm of [93] yields a subset $\mathcal{S} \subseteq \mathcal{V}$ of $\mathcal{V}$ such that each node $u \in \mathcal{S}$ has a leader $\ell(u) \in \mathcal{V}$ and the following properties hold.[1]

1. $\forall u \in \mathcal{S}: \; d(u, \ell(u)) < k$
2. $\forall u, v \in \mathcal{S}: \; \ell(u) \neq \ell(v) \implies (u, v) \notin \mathcal{E}$.
3. $\mathcal{S}$ can be computed in k rounds.
4. $\forall u \in \mathcal{V}: \; \Pr[u \in \mathcal{S}] \geq \frac{1}{en^{1/k}}$.

Thereby, $d(u, v)$ denotes the distance between two nodes u and v of $\mathcal{G}$. We apply the algorithm of [93] to obtain connected components of G with the following properties.

(I) The components have small diameter.

(II) Different components are far enough from each other such that we can define a local linear program for each component in such a way that the LPs of any two components do not interfere.

(III) Each node belongs to one of the components with probability at least p, where p depends on the diameter we allow the components to have.

Because of the limited diameter, the LPs of each component can then be computed locally. We apply the decomposition process in parallel often enough such that with high probability each node is selected a logarithmic number of times. For the decomposition of (PP) and (DP), we define linear programs (PP$'$) and (DP$'$) as follows. Let $\{y'_1, \ldots, y'_{m'}\}$ be a subset of the dual variables of (DP) and let $x'_1, \ldots, x'_{n'}$ be the primal variables which are adjacent to the given subset of the dual variables. Further let (PP$'$) and (DP$'$) be LPs where the matrix A' consists only of the columns and rows corresponding to the variables in $\underline{x}'$ and $\underline{y}'$. The relation between the original LPs (PP) and (DP) and the LPs (PP$'$) and (DP$'$) are given by the following lemma.

[1] We use $p = 1/n^{1/k}$ in the algorithm of Section 4 of [93]; the properties then directly follow from Lemma 4.1 of [93].

Algorithm 4 Covering/packing approximation in the $\mathcal{LOCAL}$ model

1: Run graph decomposition of [93] on $\mathcal{G}$;
2: **if** $v \in \mathcal{S}$ **then**
3: **send** IDs of primal neighbors to $\ell(v)$.
4: **fi**;
5: **if** $v = \ell(u)$ for some $u \in \mathcal{S}$ **then**
6: compute local PLP/DLP (cf. Lemma 2.23) of variables of $u \in \mathcal{S}$ for which $v = \ell(u)$.
7: **send** resulting values to nodes holding the respective variables.
8: **fi**

Lemma 2.23. *Every feasible solution for* (PP′) *makes the corresponding primal inequalities in* (PP) *feasible and every feasible solution for* (DP′) *is feasible for* (DP) *(variables not occurring in* (PP′) *and* (DP′) *are set to 0). Further, the values of the objective functions for the optimal solutions of* (PP′) *and* (DP′) *are smaller than or equal to the optimal values for* (PP) *and* (DP).

Proof. The feasibilities directly follow from the definition of (PP′) and (DP′). The optimal values for the objective functions of (PP′) and (DP′) are smaller than the optimal values for (PP) and (DP) because of the (DP)-feasibility of a dual feasible solution for (DP′). □

We call (PP′) and (DP′) the sub-LPs induced by the subset $\{y'_1, \dots, y'_{m'}\}$ of dual variables. We apply the graph decomposition algorithm of [93] to obtain (PP′) and (DP′) (as in Lemma 2.23) which can be solved locally.

For the decomposition of the linear program, we define $\mathcal{G}$ such that the node set $\mathcal{V}$ is the set of dual nodes of the graph G and the edge set $\mathcal{E}$ is

$$\mathcal{E} \;:=\; \big\{(u,v) \,\big|\, u,v \in \mathcal{V} \wedge \mathrm{d}_G(u,v) \leq 4\big\}\,.$$

By this, we can guarantee that non-adjacent nodes in $\mathcal{G}$ do not have neighboring primal nodes in G whose variables occur in the same constraint of (PP). Further, a message over an edge of $\mathcal{G}$ can be sent in 4 rounds on the network graph G. The basic algorithm for a dual node v to approximate (PP) and (DP) is then given by Algorithm 4.

The primal nodes only forward messages in Steps 1, 3, and 7 and receive the values for their variables in Step 7. We now take a closer look at the locally computed LPs in Line 6. By Property (II) of the graph decomposition algorithm, dual variables belonging to different local LPs cannot occur in the same dual constraint (otherwise, the according dual nodes would have to be neighbors in $\mathcal{G}$). The analogous fact holds for primal variables since dual nodes belonging to different local LPs have distance at least 6 on G and thus primal

nodes belonging to different local LPs have distance at least 4 on G. Therefore, the local LPs do not interfere and together they form the sub-LPs induced by $\mathcal{S}$ (cf. Lemma 2.23).

The complete LP approximation algorithm now consists of N independent parallel executions of the described basic algorithm. The variables of the N sub-LPs are added up and in the end, primal/dual nodes divide their variables by the maximum/minimum possible value to keep/make all constraints they occur in feasible. Therefore, the primal and dual variables x_i and y_j are divided by $\min_{j\in N_i} \frac{1}{b_j} \sum_\ell a_{j\ell} x_\ell$ and $\max_{i\in N_j} \frac{1}{c_i} \sum_\ell a_{\ell i} y_\ell$, respectively. The following theorem states how N must be chosen such that the obtained approximation ratio is optimized.

Theorem 2.24. *Let $N = \alpha e n^{1/k} \ln n$ for $\alpha \approx 4.51$. Executing the basic algorithm N times, summing up the variables of the N executions and dividing these sums as described, yields an $\alpha e n^{1/k}$ approximation of* (PP) *and* (DP) *with high probability. The algorithm needs $\mathrm{O}(k)$ rounds to complete.*

Proof. We begin the proof with the running time. The N executions can be performed completely in parallel and we therefore focus on one instance of the basic algorithm. By Property (I), the topology collecting and variable distribution in Lines 3 and 7 can be performed in $\mathrm{O}(k)$ time. The same holds for the graph decomposition in Line 1 by Property (III).

For the approximation ratio, we have to bound the ratio of the factors by which the primal and the dual variables are divided in the end. By Lemma 2.23, the dual variables of each of the N sub-LPs constitute a feasible solution for (DP). Therefore, the sums of the dual variables of the sub-LPs have to be divided by at most N to obtain a feasible solution for (DP). For the primal variables, we have to count the number of occurrences in sub-LPs for each primal constraint. This is lower-bounded by the number of times each primal node has been chosen to be in $\mathcal{S}$. By Property (IV), for each primal node the probability in each of the N executions is at least $1/(en^{1/k})$. We use Chernoff bounds to obtain an upper bound on the probability that the primal node v occurs in less than $\ln n$ sub-LPs. Let X denote the number of times v is chosen by the graph decomposition algorithm:

$$\begin{aligned} \Pr[X < \ln n] &< \left(\frac{\alpha^{1/\alpha}}{e^{1-1/\alpha}}\right)^{\alpha \ln n} = \frac{e^{\ln \alpha \ln n}}{e^{(\alpha-1)\ln n}} \\ &= \frac{1}{n^{\alpha-1-\ln\alpha}} < \frac{1}{n^2} \end{aligned}$$

for $\alpha > 4.51$. Thus, with probability at least $1 - 1/n$ all primal variables can be divided by $\ln n$. □

Corollary 2.25. *Using the network decomposition algorithm of [93], in $\mathrm{O}(k)$ rounds,* (PP) *and* (DP) *can be approximated by a factor of $\mathrm{O}(n^{1/k})$ with high probability in the $\mathcal{LOCAL}$ model. For $k \in \Theta(\log n)$, this gives a constant factor approximation in $\mathrm{O}(\log n)$ rounds.*

Algorithm 5 Distributed Randomized Rounding: Covering Problems

1: **if** $x_i \geq 1/(\lambda \ln \Delta_p)$ **then**
2: $\quad x'_i := \lceil x_i \rceil$;
3: **else**
4: $\quad p_i := x_i \cdot \lambda \ln \Delta_p$;
5: $\quad x'_i := 1$ with prob. p_i and 0 otherwise;
6: **fi**;

2.5 Randomized Rounding: Turning a Fractional into an Integer Solution

In this section, we show how to use the algorithms of Sections 2.2, 2.3, and 2.4 to solve the MDS problem or another combinatorial covering or packing problem. Hence, we show how to turn a fractional covering or packing solution into an integer one by a distributed rounding algorithm. In particular, we give an algorithm for integer covering and packing problems with matrix elements $a_{ij} \in \{0,1\}$ and where the components of the solution vectors are restricted to integers (i.e., $\underline{x}' \in \mathbb{N}^m$ and $\underline{y}' \in \mathbb{N}^n$). The solution vectors for the integer program are denoted by $\underline{x}'$ and $\underline{y}'$ whereas the solution vectors for the corresponding LPs are called $\underline{x}$ and $\underline{y}$.

We start with covering problems (problems of the form of (PP)). Because the a_{ij} and the x_i are restricted to integer values, w.l.o.g. we can round up all b_j to the next integer value. After solving/approximating the LP, each primal node v_i^p executes Algorithm 5. The value of the parameter λ will be determined later.

The expected value of the objective function is $\mathrm{E}[\underline{c}^{\mathrm{T}}\underline{x}'] \leq \lambda \ln \Delta_p \cdot \underline{c}^{\mathrm{T}}\underline{x}$. Yet regardless of how we choose λ, there remains a non-zero probability that the obtained integer solution is not feasible. To overcome this, we have to increase some of the x'_i. Assume that the j^{th} constraint is not satisfied. Let $\underline{a}_j$ be the row vector representing the j^{th} row of the matrix A and let $b'_j := b_j - \underline{a}_i\underline{x}'$ be the missing weight to make the j^{th} row feasible. Further, let $i_{j_{\min}}$ be the index of the minimum c_i for which $a_{ji} = 1$. We set $x'_{i_{j_{\min}}} := x'_{i_{j_{\min}}} + b'_j$. Applied to all non-satisfied primal constraints, this gives a feasible solution for the considered integer covering problem.

Theorem 2.26. *Let* (IP) *be an integer covering problem with* $a_{ij} = \{0,1\}$, $b_j \in \mathbb{N}$, *and* $x'_i \in \mathbb{N}$. *Furthermore, let* $\underline{x}$ *be an* α*-approximate solution for the LP relaxation* (PP) *of* (IP). *The described algorithm computes an* $\mathrm{O}(\alpha \log \Delta_p)$*-approximation* $\underline{x}'$ *for* (IP) *in a constant number of rounds.*

Proof. As stated above, the expected approximation ratio of the first part of the algorithm is $\lambda \ln \Delta_p$. In order to bound the additional weight of the second part, where $x'_{i_{j_{\min}}}$ is increased by b'_j, we define dual variables $\tilde{y}_j := b'_j c_{i_{j_{\min}}}/b_j$. For each unsatisfied primal constraint, the increase $c_{i_{j_{\min}}} b'_j$ of the primal objective

Algorithm 6 Distributed Randomized Rounding: Packing Problems

1: **if** $y_i \geq 1$ **then**
2: $\quad y'_i := \lfloor y_i \rfloor$;
3: **else**
4: $\quad p_i := 1/(2e\Delta_d)$;
5: $\quad y'_i := 1$ with probability p_i and 0 otherwise;
6: **fi**;
7: **if** $y'_i \in$ 'non-satisfied constraint' **then**
8: $\quad y'_i := \lfloor y_i \rfloor$;
9: **fi**

function is equal to the increase $b_j\tilde{y}_j$ of the dual objective function. If the j^{th} constraint is not satisfied, we have $b'_j \geq 1$. Therefore, $\mathrm{E}[\tilde{y}_j] \leq q_j c_{i_{j_{\min}}}$, where q_j is the probability that the j^{th} primal inequality is not fulfilled.

In order to get an upper bound on the probability q_j, we have to look at the sum of the x'_i before the randomized rounding step in Line 5 of the algorithm. Let $\beta_j := b_i - \underline{a}_i\underline{x'}$ be the missing weight in row j before Line 5. Because the x-values correspond to a feasible solution for the LP, the sum of the p_i involved in row j is at least $\beta_j\lambda\ln\Delta_p$. For the following analysis, we assume that $\ln\Delta_p \geq 1$. If $\ln\Delta_p < 1$, applying only the last step of the described algorithm gives a simple distributed 2-approximation for the considered integer program. Using the Chernoff bound of Theorem 1.2, we can bound q_j:

$$q_j < e^{-\frac{1}{2}\beta_j\lambda\ln\Delta_p(1-\frac{1}{\lambda\ln\Delta_p})^2} \leq \left(\frac{1}{\Delta_p}\right)^{\frac{1}{2}\lambda(1-\frac{1}{\lambda})^2} \leq \frac{1}{\Delta_p}.$$

In the second inequality, we use that $\beta_j \geq 1$. For the last inequality, we have to choose λ such that $\lambda(1-1/\lambda)^2/2 \geq 1$ (i.e., $\lambda \geq 2+\sqrt{3}$). Thus, the expected value of $\tilde{y}_j$ is $\mathrm{E}[\tilde{y}_j] \leq c_{i_{j_{\min}}}/\Delta_p$. Hence, by definition of $c_{i_{j_{\min}}}$, in expectation the $\tilde{y}$-values form a feasible solution for (DP). Therefore, the expected increase of the objective function $\underline{c}^{\mathrm{T}}\underline{x'}$ in the last step after the randomized rounding is upper-bounded by the objective function of an optimal solution for (PP). □

Combining Algorithms 3 and 5, we obtain an $\mathrm{O}(\log^2\Delta)$-approximation for MDS in the $\mathcal{CONGEST}_{\mathrm{BC}}$ model. By applying Algorithm 4 for solving the linear program ($\mathrm{LP}_{\mathrm{DS}}$), the same result is achieved in $\mathrm{O}(\log n)$ rounds in the $\mathcal{LOCAL}$ model.

We now turn our attention to integer packing problems. We have an integer program of the form of (DP) where all $a_{ij} \in \{0,1\}$ and where $\underline{y}' \in \mathbb{N}^n$. We can w.l.o.g. assume that the c_j are integers because rounding down each c_j to the next integer has no influence on the feasible region. Each dual node v_i^d applies Algorithm 6. Clearly, this yields a feasible solution for the problem. The approximation ratio of the algorithm is given by the next theorem.

Theorem 2.27. *Let* (PIP) *be an integer covering problem with* $a_{ij} = \{0,1\}$, $c_j \in \mathbb{N}$, *and* $y'_i \in \mathbb{N}$. *Furthermore, let* $\underline{y}$ *be an* α*-approximate solution for the LP relaxation of* (PIP). *The given algorithm computes an* $O(\alpha\Delta_d)$*-approximation* $\underline{y}'$ *for* (PIP) *in a constant number of rounds.*

Proof. After Line 6, the expected value of the objective function is $\underline{b}^T\underline{y}' \geq \underline{b}^T\underline{y}/(2e\Delta_d)$. We will now show that a non-zero y'_i stays non-zero with constant probability in Line 8. Let q_j be the probability that the j^{th} constraint of the integer program is not satisfied given that y'_i has been set to 1 in Line 5. For convenience, we define $Y'_j := \sum_i a_{ij}y'_i$. If $c_j \geq 2$, we apply the Chernoff bound of Theorem 1.3:

$$\begin{aligned} q_j &= \Pr\left[Y'_j > c_j \,\middle|\, y'_i = 1\right] \leq \Pr\left[Y'_j > c_j - 1\right] \\ &< \left(\frac{e^{e\Delta_d - 1}}{(e\Delta_c)^{e\Delta_d}}\right)^{c_j/(2e\Delta_d)} < \frac{1}{\Delta_d}. \end{aligned}$$

If $c_j = 1$, we get

$$\begin{aligned} q_j &\leq 1 - \Pr\left[Y'_j = 0\right] = 1 - \prod_{v_i^d \in \Gamma(v_j^p)} (1 - p_i) \\ &\leq 1 - \left(1 - \frac{1}{2e\Delta_d}\right) = \frac{1}{2e\Delta_d}. \end{aligned}$$

The probability that all dual constraints containing y'_i are satisfied is lower-bounded by the product of the probabilities for each constraint [124]. Therefore, under the natural assumption that $\Delta_d \geq 2$:

$$\Pr[y'_i = 1 \text{ after Line 8}] \geq \left(1 - \frac{1}{\Delta_d}\right)^{\Delta_d} \geq \frac{1}{4}.$$

Thus the expected value of the objective function of the integer program (PIP) is

$$\mathrm{E}[\underline{b}^T\underline{y}'] \geq 8e\Delta_d \cdot \underline{b}^T\underline{y}.$$

□

2.6 Connected Dominating Sets

One of the applications of the MDS problem which we discussed in Section 1.1.1 is clustering in ad hoc or sensor networks. In particular, we have seen that clustering helps to improve routing algorithms in such networks. However to apply clustering for routing, we usually need clusters to be connected. The resulting routing backbone then is a connected dominating set. Formally, a connected dominating set is a subset $S \subseteq V$ of the nodes of a graph $G = (V, E)$ such that S is a dominating set and such that the subgraph of G induced by

S is connected. The goal of the minimum connected dominating set (MCDS) problem is to find a connected dominating set of minimal size. In this section, we study the distributed complexity of the MCDS problem. For the $\mathcal{LOCAL}$ model, we show almost matching upper and lower bounds.

2.6.1 Upper Bound

Our distributed MCDS approximation algorithm consists of two phases. First, we compute a normal dominating set using an arbitrary distributed MDS algorithm. In the second phase, we turn the dominating set D obtained in the first phase into a connected dominating set D' by adding additional nodes to D. The following lemma shows how successful such an approach can be.

Lemma 2.28. *Let $G = (V, E)$ be a connected graph and let $\mathrm{DS}_{\mathrm{OPT}}$ and $\mathrm{CDS}_{\mathrm{OPT}}$ be the sizes of optimal dominating and connected dominating sets of G. We have*

$$\mathrm{CDS}_{\mathrm{OPT}} \;<\; 3 \cdot \mathrm{DS}_{\mathrm{OPT}}.$$

Moreover, every dominating set D of G can be turned into a connected dominating set $D' \supseteq D$ of size $|D'| < 3|D|$.

Proof. Given G and D, we define a graph $G_D = (V_D, E_D)$ as follows. The nodes of G_D are the nodes of the dominating set, that is, $V_D = D$. There is an edge $(u, v) \in E_D$ between $u, v \in D$ if and only if $d_G(u, v) \leq 3$. We first show that G_D is connected. For the sake of contradiction assume that G_D is not connected. Then there is a cut (S, T) with $S \subseteq D$, $T = D \setminus S$, and $S, T \neq \emptyset$ such that

$$\forall u \in S, \forall v \in T \;:\; d_G(u, v) \geq 4. \tag{2.15}$$

Let $u \in S$ and $v \in T$ be such that

$$d_G(u, v) \;=\; \min_{u \in S, v \in T}(d_G(u, v)). \tag{2.16}$$

By Inequality (2.15), there is a node $w \in V$ with $d_G(u, w) \geq 2$ and $d_G(v, w) \geq 2$ on each shortest path connecting u and v. Because of Equation (2.16), we have that

$$\forall u \in S, \forall v \in T \;:\; d_G(u, w) \geq 2 \wedge d_G(v, w) \geq 2.$$

However this is a contradiction to the assumption that $D = S \cup T$ is a dominating set of G.

We can now construct a connected dominating set D' as follows. We first compute a spanning tree of G_D. For each edge (u, v) of the spanning tree, we add at most two nodes such that u and v become connected. Because the number of edges of the spanning tree is $|D| - 1$, this results in a connected dominating set of size at most $3|D| - 2$. □

Algorithm 7 Computing a sparse connected subgraph

```
1: G' := G;
2: collect complete k-neighborhood;
3: for all adjacent edges e do
4:    if weight w_e of e is largest in any cycle of length ≤ 2k then
5:       remove e from G'
6:    fi
7: od
```

For the sequential case, the proof of Lemma 2.28 describes a simple algorithm to obtain a connected dominating set of size less than $3|D|$. Unfortunately, for a local, distributed algorithm, it is not possible to compute a spanning tree of G_D. Nevertheless, a similar approach also works for distributed algorithms. Instead of computing a spanning tree of G_D, it is sufficient to compute any sparse spanning subgraph of G_D. If the number of edges of the subgraph of G_D is linear in the number of nodes $|D|$ of G_D, we obtain a connected dominating set S' which is only by a constant factor larger than D.

We therefore look at the following problem. Given a graph $G = (V, E)$ with $|V| = n$, we want to compute a spanning subgraph G' of G with a minimal number of edges. For an arbitrary $k \geq 1$, Algorithm 7 shows how to compute such a spanning subgraph in k rounds in the $\mathcal{LOCAL}$ model. For the algorithm, we assume that all edges $e = (u, v)$ of G have a unique weight w_e and that there is a total order on all edge weights. If there is no natural edge weight, such a weight can for example be constructed by taking the ordered pair of the IDs of the nodes u and v. Two weights can be compared using the lexicographic order. The following lemma shows that Algorithm 7 indeed computes a sparse connected subgraph G' of G.

Lemma 2.29. *For every n-node connected graph $G = (V, E)$ and every k, Algorithm 7 computes a spanning subgraph $G' = (V, E')$ of G for which the number of edges is bounded by $|E'| \leq n^{1+O(1/k)}$.*

Proof. We first prove that the produced G' is connected. For the sake of contradiction, assume that G' is not connected. Then, there must be a cut (S, T) with $S \subseteq V$, $T = V \setminus S$, and $S, T \neq \emptyset$ such that $S \times T \cap E' = \emptyset$. However, since G is connected, there must be an edge $e \in S \times T \cap E$ crossing the given cut. Let e be the edge with minimal weight among all edges crossing the cut. Edge e can only be removed by Algorithm 7 if it has the largest weight of all edges in some cycle. However, all cycles containing e also contain another edge e' crossing the (S, T)-cut. By definition of e, $w_{e'} > w_e$ and therefore, e is not deleted by the algorithm.

Let us now look at the number of edges of G'. Because in every cycle of length at most $2k$ at least one edge is removed by Algorithm 7, G' has girth $g(G') \geq 2k + 1$. The lemma therefore follows by Theorem 1.1. □

We can therefore formulate a k-round MCDS algorithm for the $\mathcal{LOCAL}$ model consisting of the following three phases. First, a fractional dominating set is computed using Algorithm 4. Second, we use the randomized rounding scheme given by Algorithm 5 to obtain a dominating set D. Finally, Algorithm 7 is applied to G_D. For each edge (u, v) of the produced spanning subgraph of G_D, we add the nodes (at most 2) of a shortest path connecting u and v on G to D. Note that a k-round algorithm on G_D needs at most $3k$ rounds when executed on G. The achieved approximation ratio is given by the following theorem.

Theorem 2.30. *In $\mathrm{O}(k)$ rounds, the above described MCDS algorithm computes a connected dominating set of expected size*

$$\mathrm{O}\Big(\mathrm{CDS}_{\mathrm{OPT}} \cdot n^{\mathrm{O}(1/k)} \cdot \log \Delta\Big).$$

Proof. Given the dominating set D, the number of nodes of the connected dominating set D' can be bounded by

$$|D'| \;\leq\; 3|D|^{1+\mathrm{O}(1/k)} \;\leq\; 3|D|n^{\mathrm{O}(1/k)}$$

and therefore

$$\mathrm{E}\left[|D'|\right] \;\leq\; 3\mathrm{E}\left[|D|\right] n^{\mathrm{O}(1/k)}. \tag{2.17}$$

Using Theorems 2.24 and 2.26, it follows that the expected size of the dominating set D is

$$\mathrm{E}\left[|D|\right] \;\in\; \mathrm{O}\big(\mathrm{DS}_{\mathrm{OPT}} n^{\mathrm{O}(1/k)} \log \Delta\big).$$

Plugging this into Inequality (2.17) completes the proof. □

2.6.2 Lower Bound

After proving an upper bound on the time-approximation trade-off for the MCDS problem in Theorem 2.30, we will now show how to bound this trade-off from below. In particular, we will show that the described transformation of a dominating set into a connected dominating set is asymptotically optimal and that the upper bound of Theorem 2.30 is tight up to factors of $\mathrm{O}(\log \Delta)$ and $\mathrm{O}(1)$ for the approximation ratio and the time complexity, respectively. The formal statement is given by the following theorem.

Theorem 2.31. *Consider a (possibly randomized) k-round algorithm for the MCDS problem. There are graphs for which every such algorithm computes a connected dominating set S of size at least*

$$|S| \;\geq\; n^{\Omega(1/k)} \cdot \mathrm{CDS}_{\mathrm{OPT}},$$

where $\mathrm{CDS}_{\mathrm{OPT}}$ denotes the size of an optimal connected dominating set.

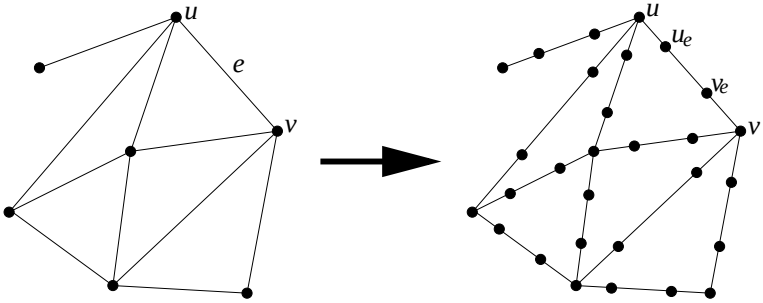

Figure 2.3: Graph transformation used for the distributed minimum connected dominating set lower bound

Proof. By Theorem 1.1, there exists a graph $G = (V, E)$ with girth $g(G) \geq (2k+1)/3$ and number of edges $|E| = n^{1+\Omega(1/k)}$, where $n = |V|$. From G, we construct a graph G' as follows. For every edge $e = (u, v) \in E$, we generate additional nodes u_e and v_e. In G', there is an edge between u and u_e, between u_e and v_e, and between v and v_e. Note that there is no edge between u and v anymore. The described transformation is illustrated by Figure 2.3. We denote the set of all new nodes by W and the number of nodes of G' by $N = |V \cup W|$.

By the definition of G', the nodes in V form a dominating set of G'. Hence, an optimal connected dominating set on G' has size less than $3|V|$ by Lemma 2.29. Note that the construction described in Lemma 2.29 actually computes a spanning tree T on G and adds all nodes $u_e, v_e \in W$ to the dominating set for which (u, v) is an edge of T. To bound the number of nodes in the connected dominating set of a distributed algorithm, we have a closer look at the locality properties of G'. Because $g(G) \geq (2k+1)/3$, the girth of G' is $g(G') \geq 2k+1$. This means that in k communication rounds it is not possible to detect a cycle of G'. Hence, no node can locally distinguish G' from a tree. However, since on a tree all edges are needed to keep a connected graph, a k-round algorithm removing an edge from G' cannot guarantee that the resulting topology remains connected. This means that the connected dominating set of every k-round algorithm must contain all nodes $V \cup W$ of G'. The approximation ratio of every distributed k-round MCDS algorithm on G' is therefore bounded by

$$\frac{|V \cup W|}{3|V|} = \frac{n^{\Omega(1/k)}}{3n} = n^{\Omega(1/k)} = N^{\left(1-\frac{1}{k+\Omega(1)}\right)\Omega(1/k)} = N^{\Omega(1/k)}.$$

□

2.7 Randomization

While discussing the $\mathcal{LOCAL}$ model in Section 1.3, we saw that randomization plays a crucial role in distributed algorithms. For many problems such as computing an MIS, there are simple and efficient randomized algorithms. For the same problems, the best deterministic algorithms are much more complicated and usually significantly slower. In this section, we discuss two randomization issues arising in distributed covering, packing and similar problems.

2.7.1 Distributed 'Derandomization'

The most important use of randomization in distributed algorithms is breaking symmetries. In Section 2.1.2, we have described that LP relaxation can be used to 'avoid' symmetry breaking. Can we also avoid randomness if we compute a fractional instead of an integer solution? In the following, we show that this is indeed the case even for general linear programs.

Assume that we are given a randomized distributed k-round algorithm $\mathcal{A}$ which computes a solution for an arbitrary linear program P. We assume that $\mathcal{A}$ explicitly solves P such that w.l.o.g. we can assume that each variable x_i of P is associated with a node v which computes x_i. We also assume that $\mathcal{A}$ always terminates with a feasible solution. The following theorem shows that $\mathcal{A}$ can be derandomized.

Theorem 2.32. *In the $\mathcal{LOCAL}$ model, algorithm $\mathcal{A}$ can be transformed into a deterministic k-round algorithm $\mathcal{A}'$ for solving P. The objective value of the solution produced by $\mathcal{A}'$ is equal to the expected objective value of the solution computed by $\mathcal{A}$.*

Proof. We first show that in the $\mathcal{LOCAL}$ model, for the node computing the value of variable x_i, it is possible to deterministically compute the expected value $\mathrm{E}[x_i]$. We have seen that in the $\mathcal{LOCAL}$ model every deterministic k-round algorithm can be formulated as follows. First, every node collects all information up to distance k. Then, each node computes its output based on this information. The same technique can also be applied for randomized algorithms. First, every node computes all its random bits. Collecting the k-neighborhood then also includes collecting the random bits of all nodes in the k-neighborhood. However, instead of computing x_i as a function of the collected information (including the random bits), we can also compute $\mathrm{E}[x_i]$ without even knowing the random bits.

In algorithm $\mathcal{A}'$, the value of each variable is now set to the computed expected value. By linearity of expectation, the objective value of $\mathcal{A}'$'s solution is equal to the expected objective value of the solution of $\mathcal{A}$. It remains to prove that the computed solution is feasible. For the sake of contradiction, assume that this is not the case. Then, there must be an inequality of P which is not satisfied. By linearity of expectation, this implies that this inequality is not

satisfied in expectation for the randomized algorithm $\mathcal{A}$. Therefore, there is a non-zero probability that $\mathcal{A}$ does not fulfill the given inequality, a contradiction to the assumption that $\mathcal{A}$ always computes a feasible solution. □

Theorem 2.32 implies that Algorithm 4 could be derandomized to deterministically compute an $O(n^{1/k})$-approximation for (PP) and (DP) in $O(k)$ rounds. It also means that in principle every distributed dominating set algorithm (e.g. [69, 118] could be turned into a deterministic fractional dominating set algorithm. Hence, when solving integer linear programs in the $\mathcal{LOCAL}$ model, randomization is only needed to break symmetries. Note that this is really a property of the $\mathcal{LOCAL}$ model and only true as long as there is no bound on message sizes and local computations. Obviously, the technique described in Theorem 2.32 can drastically increase message sizes and local computations of a randomized distributed algorithm.

2.7.2 Deterministic Symmetry Breaking

In the last section, we have seen that no randomness is needed for solving a linear program in the $\mathcal{LOCAL}$ model. However, if we need to solve an integer LP, we still have to convert the fractional solution into an integer one. This rounding is usually done by a randomized algorithm as described in Section 2.5 for covering and packing problems. Similar to the problems discussed in Section 1.3, exploring the potential of deterministic rounding algorithms is a fascinating open problem. In the non-distributed setting, randomized rounding of covering and packing problems can be derandomized using the method of conditional probabilities [104, 117, 124]. Given a (χ, d)-decomposition, these techniques can in principle be applied in the distributed setting. In time $O(\chi d)$, it is possible to deterministically achieve results similar to the ones of Theorems 2.26 and 2.27.

Taking a detailed look at deterministic symmetry breaking algorithms (e.g. [9, 33, 55, 91, 108]), it turns out that all those algorithms make extensive usage of the structure of the ID space. In fact, the number n occurring in the complexity measures of those algorithms actually represents the size of the ID space and not the number of nodes. The following observation shows that it is indeed necessary to exploit some of the structure of the ID space. Assume that nodes have unique IDs and that there is a global order on all IDs. We allow nodes to compare two IDs with respect to this order, however, we assume that nodes are not able to perform any other operation involving IDs. Given such a strong model, it is not possible to deterministically break symmetries. As an example, consider computing an MIS of a ring. Note that if IDs are numbers from 1 to N, we can deterministically compute such an MIS in time $O(\log^* N)$ [33]. However, if we can only compare IDs, it is not possible to deterministically compute an MIS on a ring in a small number of rounds. Assume that except for the smallest ID which follows the largest one, the nodes of the ring are labeled in increasing order. Each k-neighborhood which does not contain the largest and

the smallest ID consists of an increasing sequence of IDs. Therefore, for small enough k, in a k-round algorithm, almost all nodes have the same local view. In a deterministic algorithm, they all have to decide in the same way which is not possible if they have to compute an MIS. A similar argument also works for MDS approximation algorithms. Instead of a regular ring, we consider a ring where every node is adjacent to the next δ nodes to the left and to the right. Again, if IDs are assigned in increasing order, almost all nodes have the same view resulting in a dominating set consisting of almost all nodes.

Chapter 3

Lower Bounds for Distributed Problems

After exploring upper bounds for covering and packing problems in Chapter 2, the goal of this chapter is to find lower bounds on the possible time approximation trade-offs for covering and packing problems. While there are many algorithms providing upper bounds for a variety of distributed problems, almost nothing is known about lower bounds, especially for the $\mathcal{LOCAL}$ model. In this chapter, we prove lower bounds for several traditional graph theory problems.

A pioneering and seminal lower bound by Linial [91] shows that the non-uniform $O(\log^* n)$ coloring algorithm by Cole and Vishkin [33] is asymptotically optimal for the ring. For different models of distributed computation, there is a number of other lower bounds [44, 85], most notably for the problem of constructing a minimum spanning tree of the network graph [39, 38, 94, 113]. With the exception of [39], the MST lower bounds apply to the $\mathcal{CONGEST}$ model. In [38], the lower bounds of [94, 113] are extended to approximation algorithms. To the best of our knowledge, it is the only previous lower bound on distributed hardness of approximation.

Chapter 3 is organized as follows. In Section 3.1, we prove a lower bound for 2-round MDS algorithms to introduce the general idea behind all presented lower bounds. In Section 3.2, the ideas of 3.1 are extended to an arbitrary number of rounds but for a simpler problem. For every k, we give a lower bound on the possible approximation ratio of a k-round minimum vertex cover algorithm. Finally, Section 3.3 shows that the obtained MVC lower bound can be extended to a whole set of optimization problems including minimum dominating set and maximum matching. Moreover, the lower bound of Section 3.2 even allows to give new strong time lower bounds for the distributed construction of maximal matchings and maximal independent sets. Thus, all presented lower bounds hold for the $\mathcal{LOCAL}$ model. While for upper bounds, this is weakest possible

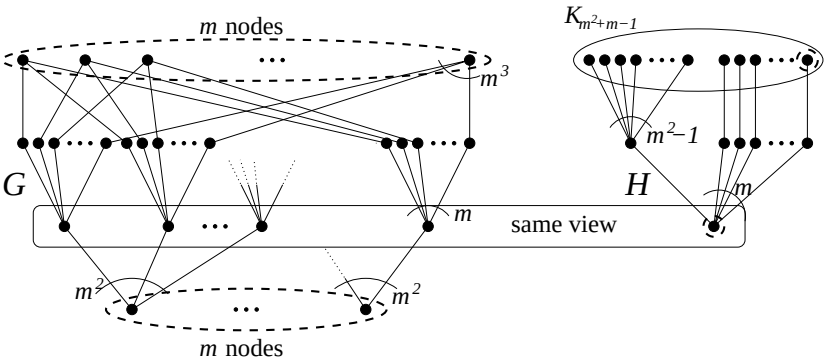

Figure 3.1: Lower bound graphs G and H for 2-round MDS algorithms. All nodes on Layer II have the same 2-hop view. Optimal dominating sets are marked by dashed lines.

model, for lower bound, the $\mathcal{LOCAL}$ model is the strongest possible model. All lower bounds are a true consequence of locality limitations, and not merely a side-effect of congestion, asynchrony, or limited message size.

3.1 Two-Round Dominating Set Lower Bound

The proofs of the 2-round MDS lower bound and of the MVC lower bound of the next section are based on the timeless indistinguishability argument [45, 87]. 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. In particular for the MDS lower bound, we construct two graphs G and H such that after 2 communication rounds, there is a set of nodes S of G and a node v of H which see exactly the same graph topology. Informally speaking, all of them base their decision on the same information and thus all have to make the same decision. Either all those nodes join the dominating set or all of them decide not to join the dominating set. However, both decisions are ruinous because G has a dominating set which is much smaller than $|S|$ and because H has no small dominating set not containing v.

The graphs G and H are displayed in Figure 3.1. Both graphs depend on a parameter m which can have any integer value greater than 1. Graph G consists of 4 layers which we number from I to IV where Layer I is the bottom layer in Figure 3.1. Layer I consists of m nodes. Each of them is connected to m^2 distinct nodes on Layer II, that is, there are m^3 nodes in Layer II. Similarly, each of the m^3 nodes in Layer II is connected to m distinct nodes in Layer III. The number of nodes of G in Layer III is therefore m^4. We denote the nodes in

Layer II by $v_1, \ldots, v_{m^3}$ and the nodes on Layer III by v_{ij} where $i = 1, \ldots, m^3$ and $j = 1, \ldots, m$. The neighbors of node v_i on Layer II are called $v_{i1}, \ldots, v_{im}$. On the top Layer IV, there are m nodes $u_1, \ldots, u_m$. Node u_j is connected to the j^{th} Layer III neighbor of every Layer II node, that is, node u_i is adjacent to v_{ij} for $i = 1, \ldots, m^3$. Let us now look at the view of Layer II nodes after 2 communication rounds. In Section 1.2, we defined the (unlabeled) view after 2 rounds of a node v as the graph induced by all nodes in $\Gamma_2^+(v)$ except for edges between nodes at distance exactly 2. In G, node v_i from Layer II has m neighbors v_{ij} on Layer III and one neighbor on Layer I. For each of the Layer III neighbors, v_i sees one Layer IV node at distance 2, whereas v_i's neighbor on Layer I has $m^2 - 1$ neighbors different from v_i. Therefore, v_i sees a tree of depth 2. The root v_i has $m+1$ neighbors, one of them has $m^2 - 1$ children, the others have 1 child. For the second graph H, we start with the tree describing the view of node v_i. We place one node w in Layer II and $m+1$ adjacent nodes in Layer III. One of the adjacent nodes has $m^2 - 1$ neighbors in Layer IV, the other m nodes have one neighbor in Layer IV. The Layer IV nodes are connected to a clique of size $m^2 + m - 1$. Note that because all the edges of the clique are between nodes at distance exactly 2 from w, all nodes on Layer II see the same topology. Based on this observation, we are able to prove the following theorem

Theorem 3.1. *Let $\mathcal{A}$ be a 2-round distributed and possibly randomized MDS algorithm. There is a constant c such that for every $n \geq c$, there is a graph with n nodes on which the approximation ratio of $\mathcal{A}$ is at least $\Omega(\sqrt{n})$.*

Proof. We have already shown that all Layer II nodes of G and H see the same tree topology T. However, we have not yet considered the effects of node identifiers and randomization. For each labeled instance $\mathcal{L}(T)$ of T, there is some probability $P(\mathcal{L}(T))$ that a node with 2-round view $\mathcal{L}(T)$ joins the dominating set in algorithm $\mathcal{A}$. Let $\mathcal{L}_0(T)$ be the labeling which minimizes this probability and let $p := P(\mathcal{L}_0(T))$.

We first look at the approximation ratio of $\mathcal{A}$ for graph H. An optimal dominating set of H consists of the Layer II node w and a node from the clique on Layer IV. Hence, a minimum dominating set of H has size 2. Assume that w does decide not to join the dominating set. In this case, each of the $m+1$ neighbors of w has to be covered by a different node. Therefore, the smallest dominating set of H which does not contain w has size $m+1$. Assume now that the view of Layer II node w is $\mathcal{L}_0(T)$, that is, w joins the dominating set with probability p. Let $n_H \in \mathrm{O}(m^2)$ be the number of nodes of H. The expected size of the dominating set produced by $\mathcal{A}$ is therefore lower bounded by

$$\mathrm{E}\left[\mathrm{DS}_{\mathcal{A}}(H)\right] \;\geq\; 2p + (1-p)(m+1) \;\in\; \Omega\big(p + (1-p)\sqrt{n_H}\big). \tag{3.1}$$

Let us now look at the approximation ratio of $\mathcal{A}$ for G. An optimal dominating set of G consists of the $2m$ nodes of Layers I and IV. Because p is the minimum of the probabilities $P(\mathcal{L}(T))$, each of the m^3 nodes of Layer II joins

the dominating set with probability at least p in algorithm $\mathcal{A}$. Analogously to above, let $n_G \in \mathrm{O}(m^4)$ be the number of nodes of G. By linearity of expectation, we have

$$\mathrm{E}\left[\mathrm{DS}_{\mathcal{A}}(G)\right] \;\geq\; (1-p)2m + pm^3 \;\in\; \Omega\big((1-p)n_G^{1/4} + pn_G^{3/4}\big). \tag{3.2}$$

No matter how we choose $p \in [0,1]$, the approximation ratio for either G or H is bad. If p is bounded away from 0 by a constant, by Inequality (3.2), the approximation ratio of $\mathcal{A}$ on G is at least $\Omega(m^3/m) = \Omega(n_G^{3/4}/n_G^{1/4}) = \Omega(\sqrt{n_G})$. If p is bounded away from 1 by a constant, by Inequality (3.1), the approximation of $\mathcal{A}$ on H is at least $\Omega(m^2/1) = \Omega(\sqrt{n_H})$. □

3.2 Lower Bound for Minimum Vertex Cover

Unfortunately, the MDS lower bound of the previous section cannot directly be extended to more than 2 rounds. To find a similar lower bound for an arbitrary number of rounds, we look at a simpler but related problem. A vertex cover for a graph $G = (V, E)$ is a subset of nodes $V' \subseteq V$ such that for each edge $(u, v) \in E$, at least one of the two incident nodes u, v belongs to V'. Finding a vertex cover with minimum cardinality is known as the MVC problem.

Intuitively MVC could be considered 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 this decision. Symmetry breaking as introduced in Chapters 1 and 2 is not necessary for the MVC problem. A solution of the covering LP describing the fractional variant of MVC can be turned into a vertex cover by rounding up all variables whose value is at least 1/2. Therefore a fractional solution can be turned into an integer one without communication. The size of the resulting vertex cover is at most by a factor of 2 larger than the LP solution. For δ-regular graphs, finding a constant approximation for MVC is particularly simple. Because every node can cover at most δ edges, we need at least $n/2$ nodes to cover all $n\delta/2$ edges. Taking all nodes therefore gives a 2-approximation. The fact that there is a simple greedy algorithm which approximates MVC within a factor of 2 in the global setting, additionally raises hope for an efficient local algorithm.

In the following, we show that nevertheless, MVC cannot be approximated arbitrarily well by a local algorithm. Similar to the 2-round MDS lower bound, we show that in a constant number of rounds, the approximation ratio cannot be better than a constant root of n or Δ. The special properties of MVC even imply that the obtained lower bounds are in fact hardness results for fractional covering problems whereas a similar lower bound for MDS would only give a bound for integer problems.

We first give an outline of the proof. The basic idea is to construct a graph $G_k = (V, E)$, for each positive integer k. We show that after k communication

rounds, two neighboring nodes see exactly the same graph topology; informally speaking, both neighbors are equally qualified to join the vertex cover. However, in our example graph G_k, choosing the wrong neighbor will be ruinous.

Graph 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 3.2. 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 $v \in S$ 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, we will use C_0 and C_1 to denote the two sets of the bipartite subgraph S.

Our proof is organized as follows. The structure of G_k is defined in Subsection 3.2.1. In Subsection 3.2.2, we show how G_k can be constructed without small cycles, ensuring that each node sees a tree within distance k. Subsection 3.2.3 proves that adjacent nodes in C_0 and C_1 have the same view $\mathcal{T}_{v,k}$ and finally, Subsection 3.2.4 derives the lower bounds.

3.2.1 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*.

We define the structure of G_k using a directed tree $CT_k = (\mathcal{C}, \mathcal{A})$ with doubly labeled 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 a *leaf-cluster* if it is adjacent to only one other cluster, and we call it an *inner-cluster* otherwise.

Definition 3.1. *The cluster tree CT_k is recursively defined as follows:*

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

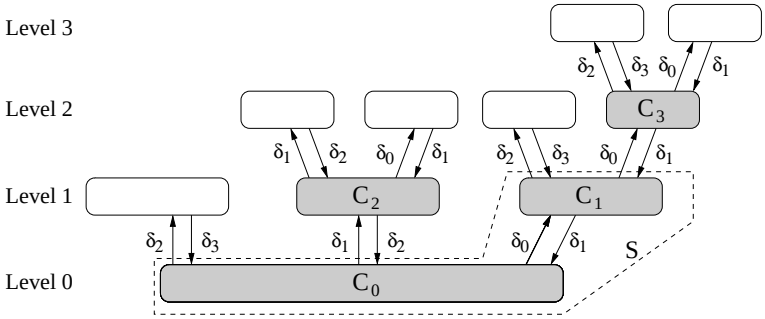

Figure 3.2: Cluster-Tree CT_2.

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 3.2 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 3.2). The *depth* of a cluster C is its distance 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 3.2, 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 view is a tree.

3.2.2 The Lower Bound Graph

In Subsection 3.2.3, 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 [88]. 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 the interested reader, we refer to [88].

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 labeled by the r-vectors over the finite field $\mathbb{F}_q$, i.e. $P = L = \mathbb{F}_q^r$. In accordance with [88], 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):

$$\begin{aligned}(p) &= (p_1, p_{1,1}, p_{1,2}, p_{2,1}, p_{2,2}, p'_{2,2}, p_{2,3}, p_{3,2}, \dots \\ &\quad p_{i,i}, p'_{i,i}, p_{i,i+1}, p_{i+1,i}, \dots) \qquad (3.3)\\ [l] &= [l_1, l_{1,1}, l_{1,2}, l_{2,1}, l_{2,2}, l'_{2,2}, l_{2,3}, l_{3,2}, \dots \\ &\quad l_{i,i}, l'_{i,i}, l_{i,i+1}, l_{i+1,i}, \dots]. \qquad (3.4)\end{aligned}$$

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 $[l]$ if and only if the first $r-1$ of the following conditions hold (for $i = 2, 3, \dots$).

$$\begin{aligned} l_{1,1} - p_{1,1} &= l_1 p_1 \\ l_{1,2} - p_{1,2} &= l_{1,1} p_1 \\ l_{2,1} - p_{2,1} &= l_1 p_{1,1} \\ l_{i,i} - p_{i,i} &= l_1 p_{i-1,i} \\ l'_{i,i} - p'_{i,i} &= l_{i,i-1} p_1 \\ l_{i,i+1} - p_{i,i+1} &= l_{i,i} p_1 \\ l_{i+1,i} - p_{i+1,i} &= l_1 p'_{i,i} \end{aligned} \qquad (3.5)$$

In [88], 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 is fixed, the remaining coordinates of v are uniquely determined. This is concretized in the next lemma.

Lemma 3.2. *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 Equation (3.5) 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 3.2.4. 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 3.3. *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 3.2, 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 [89], 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 our asymptotic results remain unaffected by this observation, we continue to use G_k as constructed above.

3.2.3 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 3.2. *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 3.3. *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 3.3.

Example 3.1. *Consider G_1. Let V_{C_0} and V_{C_1} denote the view-trees of nodes in C_0 and C_1, respectively:*

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

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.

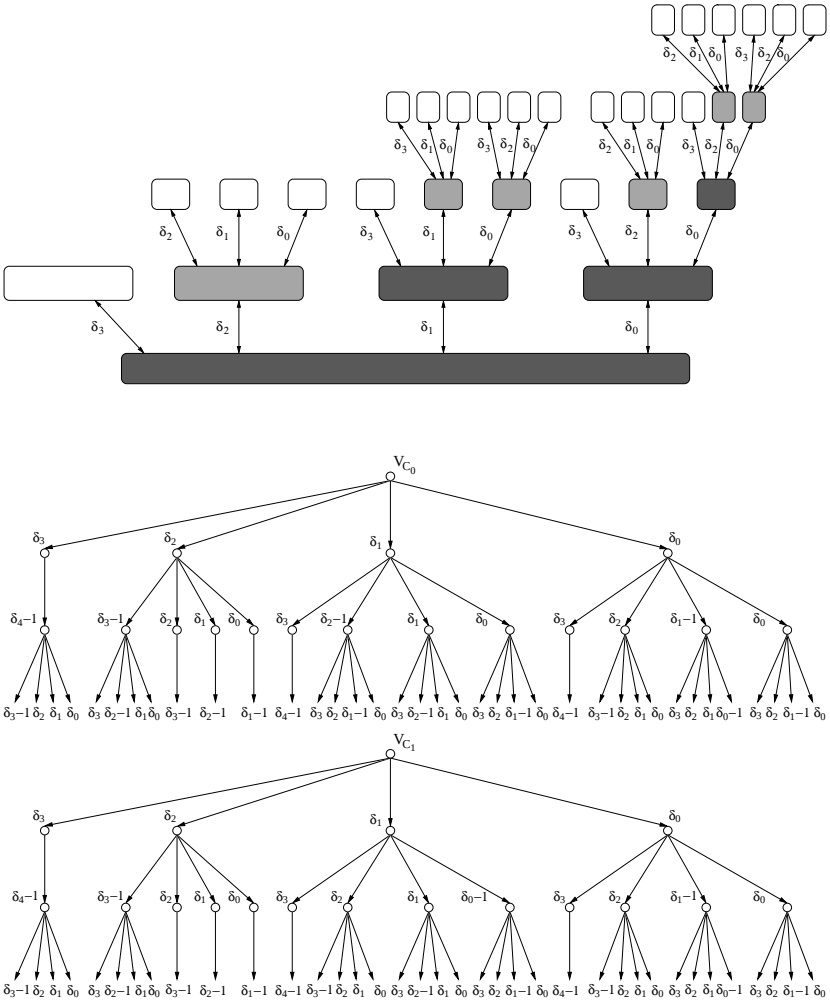

Figure 3.3: The Cluster Tree CT_3 and the corresponding view-trees of nodes in C_0 and C_1. The cluster trees CT_1 and CT_2 are shaded dark and light, respectively. The labels of the arcs of the cluster tree represent the number of neighbors of nodes of the lower-level cluster in the neighboring higher-level cluster. The labels of the reverse links are omitted. In the view-trees, an arc labeled with δ_i stands for δ_i edges, all connecting to identical subtrees.

Lemma 3.4. *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 3.1 that the rule holds for $k = 1$. For the induction step, we build CT_{k+1} from CT_k as defined in Definition 3.1. $M^{(k)}$ 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 for $r < k$. 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 3.1, 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 came 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 3.4. *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,*

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

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

Using the notion of r-equality, it is now easy to define what we actually have to prove. We will show that in G_k, $V_{C_0} \overset{k}{=} V_{C_1}$ 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 will now establish several helper lemmas.

Lemma 3.5. *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*

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

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,

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

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

Lemma 3.6. *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 $r \leq \min(d, d')$,*

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

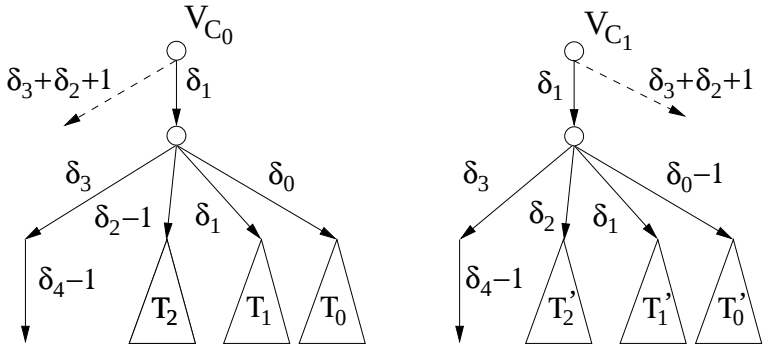

Figure 3.4: The view-trees V_{C_0} and V_{C_1} in G_3 seen upon using link δ_1.

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 3.1, 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 3.5 concludes the proof. □

Figure 3.4 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 3.7. *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 $i \in \{2, \ldots, k\}$,

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

Proof. Again, we make use of Lemma 3.4 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.

$$\begin{aligned} M_i &= \bigcup_{\substack{j=0\ldots k \\ j\neq i-1}} B^{\uparrow}_{j,k-j,1} \cup B^{\uparrow,\neg}_{i-1,k-i+1,1} \\ B^{\uparrow}_{i-2,k-i,2} &= \bigcup_{\substack{j=0\ldots i+1 \\ j\neq i-1}} B^{\uparrow}_{j,k-i-1,3} \cup \bigcup_{j=i+2\ldots k} B^{\uparrow}_{j,k-j,3} \\ &\quad \cup \; B^{\downarrow,\neg}_{i-1,k-i+1,1} \end{aligned}$$

For $j = \{0, \ldots, i-2, i, \ldots, k\}$, all subtrees are equal according to Lemmas 3.5 and 3.6. 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 3.4 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, we have to show $M_i \overset{k-i-3}{=} B^{\uparrow}_{i-2,k-i,2}$. 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 3.8. *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):

$$V_{C_0} \quad : \quad M_* = \bigcup_{j=0\ldots k} B^{\uparrow}_{j,k-j,1}$$

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

It follows that $V_{C_0} \overset{k}{=} V_{C_1} \impliedby B^{\uparrow}_{1,k-1,1} \overset{k-1}{=} M_1$ because all other subtrees are $(k-1)$-equal by Lemma 3.5. 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 $M_2 \overset{k-2}{=} B^{\uparrow}_{2,k-2,1}$:

$$\begin{aligned} M_1 &= \bigcup_{j=1\ldots k} B^{\uparrow}_{j,k-j,1} \cup\ B^{\uparrow,\neg}_{0,k,1} \\ B^{\uparrow}_{1,k-1,1} &= B^{\uparrow}_{0,k-2,2} \cup B^{\uparrow}_{1,k-2,2} \cup D_{\{\}} \cup\ M_2^{\neg} \\ &\overset{k-2}{=}_{\text{Lem. 3.7}} B^{\uparrow,\neg}_{0,k-2,2} \cup B^{\uparrow}_{1,k-2,2} \cup D_{\{\}} \cup\ M_2. \end{aligned}$$

By Lemmas 3.5 and 3.6, all subtrees 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 $V_{C_0} \overset{k}{=} V_{C_1} \impliedby B^{\uparrow}_{r+1,k-r-1,1} \overset{k-r-1}{=} M_{r+1}$.

$$\begin{aligned} M_r &= \bigcup_{\substack{j=0\ldots k \\ j\neq r-1}} B^{\uparrow}_{j,k-j,1} \cup\ B^{\uparrow,\neg}_{r-1,k-r+1,1} \\ B^{\uparrow}_{r,k-r,1} &= \bigcup_{j=0\ldots r} B^{\uparrow}_{j,k-r-1,2} \cup D_{\{\}} \cup\ M^{\neg}_{r+1} \\ &\overset{k-r-1}{=}_{\text{Lem. 3.7}} \bigcup_{\substack{j=0\ldots r \\ j\neq r-1}} B^{\uparrow}_{j,k-r-1,2} \cup B^{\uparrow,\neg}_{r-1,k-r-1,2} \\ &\quad \cup \bigcup_{j=r+2\ldots k} B^{\uparrow}_{j,k-j,2} \cup\ M_{r+1}. \end{aligned}$$

Apart from M_{r+1} (resp,. $B^{\uparrow}_{r+1,k-r-1,1}$), all subtrees are $(k-r-1)$-equal by Lemmas 3.5 and 3.6. 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 3.8 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.

3.2.4 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 3.9. *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 3.2.2 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 labeling $\mathcal{L}(\mathcal{T}_{v,k})$. Assume that the labeling 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 labeling. By Theorem 3.8, 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 labelings $\mathcal{L}(\mathcal{T}_{v_0,k})$ 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 labeling for which at least half of the nodes of C_0 are in the vertex cover.[1]

The argument for randomized algorithms is now straightforward 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

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

optimal deterministic algorithm for an arbitrarily chosen distribution on the labels. An alternative proof for randomized algorithm is by Theorem 2.32. Because for MVC, solving the linear program relaxation is up to a factor of 2 equivalent to solving MVC, every randomized MVC algorithm can be derandomized at the cost of at most a factor of 2. □

Lemma 3.9 gives a lower bound on the number of nodes chosen by any k-local MVC algorithm. In particular, we have that $\mathrm{E}\left[|ALG|\right] \geq |C_0|/2 = n_0/2$. 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

$$\delta_i := \delta^i \quad , \forall i \in \{0, \ldots, k+1\} \tag{3.6}$$

for some value δ.

Lemma 3.10. *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 Equation (3.6), the number of nodes per cluster decreases for each additional level by a factor of 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

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

□

It remains to determine the relationship between δ and n_0 such that G_k can be realized as described in Subsection 3.2.2. There, the construction of G_k with large girth is based on a smaller instance G'_k where girth does not matter. Using Equation (3.6) (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 of 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 3.10. Applying the construction of Subsection 3.2.2, 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 everything together, we get

$$n_0 \;\leq\; (4n'_0)^{2k-4} \;\leq\; 4^{2k-4}\delta^{4k^2}. \tag{3.7}$$

Theorem 3.11. *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*

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

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

Proof. We can choose $\delta \geq 4^{-1/(2k)} n_0^{1/(4k^2)}$ due to Inequality (3.7). Finally, using Lemmas 3.9 and 3.10, the approximation ratio α is at least

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

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

Theorem 3.12. *In order to obtain a poly-logarithmic or even constant approximation ratio, every distributed algorithm for the MVC problem requires at least*

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

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 3.11, 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 poly-logarithmic 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 $k = \beta\log\Delta/\log\log\Delta$. □

Having proven those distributed lower bounds for the MVC problem, it is of course interesting to compare the obtained results with the best known distributed MVC algorithm. MVC is a classical covering problem and can therefore be solved using Algorithm 3. Because each edge can only be covered by two nodes, the degree of each dual node in the network graph for Algorithm 3 is 2. By Theorem 2.21, we can therefore compute a $\Delta^{\mathrm{O}(1)/k}$-approximation in k rounds using Algorithm 3. Note that all coefficients of the LP underlying MVC are either 0 or 1. For $k \in \mathrm{O}(\log\Delta/\log\log\Delta)$, we have $\Delta^{1/k}/k = \Delta^{\mathrm{O}(1)/k'}$ for $k' \in \Theta(k)$. Therefore, Theorem 3.11 is tight in this case. By Theorem 3.12, the gap between lower and upper bound is $\mathrm{O}(\log\log\Delta)$ for constant-factor approximations. Note that the lower bound is given for the $\mathcal{LOCAL}$ model whereas the upper bound works in the $\mathcal{CONGEST}_{\mathrm{BC}}$ model. As a consequence, we have proven that up to the small gap between upper and lower bound, the complexity of MVC is the same for the $\mathcal{LOCAL}$ and the $\mathcal{CONGEST}_{\mathrm{BC}}$ model.

For the MVC lower bound which is expressed as a function of the number of nodes n, Theorem 3.11 is not nearly as tight. The approximation ratio of Algorithm 3 can be $\Delta^{\Theta(1)/k}$ even if $\Delta = \Theta(n)$. The upper bound therefore deviates from the lower bound by a factor of $\Theta(\sqrt{\log n\log\log n})$ for constant-factor approximations. The reason for this larger gap is the large girth of G_k. Intuitively, in order to be able to show a large MVC approximation lower bound with the described technique, most of the nodes of a bad graph have to be 'bad' nodes. However, if those bad nodes all have degree δ, the graph needs to have at least $n^{\Omega(k)}$ nodes in order to have girth $\Omega(k)$. However, by taking all non-'bad' nodes and applying Algorithm 3 to the bad nodes, we obtain an $\mathrm{O}(\delta^{\mathrm{O}(1)/k})$-approximation in k rounds. An $\Omega(n^{\Omega(1)/k^2})$ lower bound is the best we can hope for in this case. It therefore seems that we either need a different technique or we need to be able to handle graphs with small girth in order to significantly improve the $\Omega(n^{c/k^2}/k)$-lower bound.

3.3 Extending the Lower Bound to Other Problems

3.3.1 Minimum Dominating Set

In a non-distributed setting, minimum dominating set is equivalent to the general minimum set cover problem[2] whereas MVC is a special case of set cover which can be approximated much better. It is therefore not surprising that in a distributed environment, MDS is strictly harder than MVC, too. In the following, we show that this intuitive fact can be formalized.

Theorem 3.13. *There are graphs G, such that in k communication rounds, every 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 an MDS instance with the same locality. Let $G' = (V', E')$ be a graph for which we want to solve MVC. We construct the corresponding dominating set graph $G = (V, E)$ 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. Clearly, the localities of G' and G are the same, 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 3.11. □

[2] There exist approximation preserving reductions in both directions.

Corollary 3.14. *To obtain a poly-logarithmic or constant approximation ratio for minimum dominating set, there are graphs on which every distributed algorithm needs time*

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

Proof. The corollary is a direct consequence of Theorem 3.13 and the proof of Theorem 3.12. □

Remark: Using exactly the same reduction as for MDS, we can also show that solving minimum fractional dominating set ($\mathrm{LP_{DS}}$) is as hard as solving minimum fractional vertex cover. Therefore, Theorem 3.13 and Corollary 3.14 also hold for the fractional dominating set problem.

3.3.2 Maximum Matching

We will now show that the MVC lower bound can also be extended to packing problems. In particular, we show that for maximum fractional matching—and therefore also for maximum matching—, the same lower bounds hold. A matching is a set M of edges such that no two edges in M are adjacent. The fractional version is the natural LP relaxation as defined by the following linear program.

$$\begin{aligned} \max \quad & \sum_{e_j \in E}^{n} y_j \\ \text{subject to} \quad & \sum_{v_j \in E(v)} y_j \leq 1, \forall v \in V \\ & y_j \geq 0. \end{aligned} \tag{$\mathrm{LP_M}$}$$

Thereby, $E(v)$ denotes the set of edges incident to node v. Note that ($\mathrm{LP_M}$) is the LP dual of the fractional vertex cover problem. The approach to prove a lower bound for ($\mathrm{LP_M}$) is similar to the approach for the MVC lower bound. For every graph, we construct a graph H_k containing a large set S of edges all having the same view up to distance k. The view of two edges (u, v) and (u', v') is defined to be equal if $\mathcal{V}_{u,k} \cup \mathcal{V}_{v,k} = \mathcal{V}_{u',k} \cup \mathcal{V}_{v',k}$. The graph H_k is constructed such that some of the edges in S need to have a large y-value to achieve a good ($\mathrm{LP_M}$)-solution whereas other edges in S must not have a large y-value to obtain a feasible ($\mathrm{LP_M}$)-solution.

The construction of H_k is based on the construction of the MVC lower bound graph G_k as follows. Let G_k and G'_k be two identical copies of the MVC lower bound graph and let $\varphi : V(G_k) \to V(G'_k)$ be a graph isomorphism mapping nodes of G_k to nodes of G'_k. The graph H_k consists of G_k and G'_k and additional edges between nodes $v \in G_k$ and $v' \in G'_k$ if and only if $v' = \varphi(v)$.

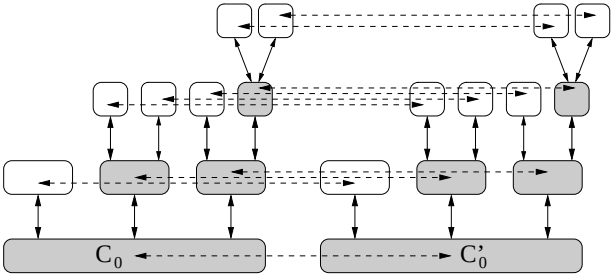

Figure 3.5: Lower Bound Graph H_k

Hence, each node of G_k is adjacent to its counterpart in G'_k. The set of nodes of G'_k corresponding to cluster C_i in G_k is called cluster C'_i. The structure of H_k is illustrated in Figure 3.5. Similarly to G_k where all nodes in clusters C_0 and C_1 have the same local view (Theorem 3.8), all nodes of clusters C_0, C_1, C'_0, and C'_1 have the same distance k view in H_k.

Lemma 3.15. *Let u and v be two nodes from any of the clusters C_0, C_1, C'_0, and C'_1 of H_k. Up to distance k, u and v see the same topology, that is, $\mathcal{T}_{u,k} = \mathcal{T}_{v,k}$.*

Proof. Immediate from the definition of H_k and from Theorem 3.8. □

As a consequence of Lemma 3.15, no k-round algorithm can distinguish between edges connecting two nodes in $C_0 \cup C_1 \cup C'_0 \cup C'_1$. In particular, edges between C_0 and C_1 can not be distinguished from edges between C_0 and C'_0. Based on this observation, we can now derive a lower bound on the approximation ratio for k-local ($\mathrm{LP_M}$)-algorithms. Let OPT be the optimal solution for ($\mathrm{LP_M}$) and let ALG be the solution computed by any algorithm. Further let $S_0 := C_0 \cup C'_0$ and let $S_1 := C_1 \cup C'_1$.

Lemma 3.16. *When applied to $H_k = (V, E)$ as constructed in Subsection 3.2.2, any distributed, possibly randomized algorithm which runs for at most k rounds computes, in expectation, an ($\mathrm{LP_M}$)-solution of size at most*

$$|ALG| \;\leq\; \frac{|S_0|}{2\delta^2} + \big(|V| - |S_0|\big).$$

Proof. We first consider deterministic algorithms. The fractional value assigned to $e_i = (u, v)$ by an algorithm is denoted by y_i. The decision of which value y_i is assigned to edge e_i depends only on the topologies $\mathcal{T}_{u,k}$ and $\mathcal{T}_{v,k}$ and on 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 an arbitrary node u is chosen uniformly at random, too.

For convenience, we define the view of an edge $e_i = (u, v)$ as the union of the views of u and v. All edges connecting nodes in S_0 and S_1 see the same topology. If the labels are chosen uniformly at random, it follows that the distribution of the views and therefore the distribution of the y_i is the same for all those edges. We call the random variables describing the distribution of the y_i, Y_i. Let $u \in S_1$ be a node of S_1. Node u has δ^2 neighbors in S_0. Therefore, for edges e_i between nodes in S_0 and S_1, by linearity of expectation, $\mathrm{E}[Y_i] \leq 1/\delta^2$ because otherwise there exist labelings for which the calculated solution is not feasible. By Lemma 3.15, edges e_j connecting nodes in C_0 and C_0' have the same view as edges between S_0 and S_1. Hence, also for the value y_j of e_j, $\mathrm{E}[Y_j] \leq 1/\delta^2$ must hold. There are $|S_0|/2$ such edges and therefore the expected total value contributed by edges between two nodes in S_0 is at most $|S_0|/(2\delta^2)$.

All edges which do not connect two nodes in S_0, have one end-point in $V \setminus S_0$. In order to get a feasible solution, the total value of all edges adjacent to a set of nodes V' can be at most $|V'|$. This can for example be seen by looking at the dual problem, a kind of minimum vertex cover where some edges only have one end node. Taking all nodes of V' (assigning 1 to the respective variables) yields a feasible solution for this vertex cover problem. The lemma therefore holds for deterministic algorithms.

For randomized algorithms, we can use the same arguments as in the MVC lower bound. The claim follows by either applying Yao's minimax principle or Theorem 2.32 which states that distributed LP algorithms can be derandomized. □

We now derive the lower bound. Lemma 3.16 gives an upper bound on the number of nodes chosen by any k-local ($\mathrm{LP_M}$)-algorithm. Choosing all edges connecting nodes of G_k and G_k' is feasible, hence, $|OPT| \geq n/2$. Let α denote the approximation ratio achieved by a k-round algorithm. As in the MVC proof we assume that $k + 1 \leq \delta/2$. By Lemma 3.10, we have

$$\alpha \;\geq\; \frac{n}{|S_0|\left(\frac{1}{2\delta^2} + \frac{k+1}{\delta-(k+1)}\right)} \;\geq\; \frac{|S_0|}{|S_0|\left(\frac{1}{\delta} + \frac{2(k+1)}{\delta}\right)} \;=\; \frac{\delta}{2k+3}.$$

The following theorem is now obtained in analogy to Theorem 3.11.

Theorem 3.17. *There are graphs G, such that in k communication rounds, every distributed possibly randomized 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.

Corollary 3.18. *In order to obtain a poly-logarithmic or even constant approximation ratio, every distributed algorithm for the linear program ($\mathrm{LP_M}$) 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.

3.3.3 Maximal Independent Sets and Matchings

Recall that a maximal matching (MM) of a graph G is a maximal set of edges which do not share common end-points. Hence, an MM is a set of non-adjacent edges of G such that all edges in $E(G) \setminus MM$ have a common end-point with an edge in MM. A maximal independent set (MIS) is a maximal set of non-adjacent nodes, that is, all nodes not in the MIS are adjacent to some node of the MIS. The best known lower bound for the distributed computation of an MM or an MIS is $\Omega(\log^* n)$ which holds for rings [91]. Based on Theorem 3.12, we get the following stronger lower bounds.

Theorem 3.19. *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 matching. The same lower bounds hold for the construction of maximal independent sets.

Proof. It is well known that the set of all end-points of the edges of an MM 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 an MM therefore directly follows from Theorem 3.12.

Theorem 3.12 does not directly imply a lower bound for randomized MM algorithms. Theorem 3.12 gives a lower bound on the approximation ratio which can be achieved in time exactly k. It does not say anything about algorithms where the time complexity is at most k in expectation or with a certain probability. However, when considering randomized algorithms for constructing an MM, we are interested in algorithms which always compute a correct solution and where only the time complexity depends on randomness. In other words, Theorem 3.12 bounds the potential of Monte Carlo type algorithms whereas for constructing an MM, we are interested in Las Vegas type algorithms.

To extend the proof towards randomized algorithms, we show how to convert any distributed MM algorithm with expected time complexity t into a vertex cover algorithm with fixed time complexity $2t + 1$ and expected approximation ratio $\mathrm{O}(\log \Delta)$. Assume that we are given a randomized MM algorithm $\mathcal{A}$ which

terminates after t rounds in expectation. When running $\mathcal{A}$ for $2t$ rounds, by the Markov inequality, the probability for having an MM is at least $1/2$. We can therefore modify $\mathcal{A}$ to obtain a $2t$-round algorithm $\mathcal{A}'$ computing a matching M which is maximal with probability at least $1/2$. Let $S \subseteq V$ be the set consisting of all nodes adjacent to an edge of M. With probability at least $1/2$, S is a vertex cover. In any case, the size $|S|$ of S is at most twice the size of an optimal vertex cover. If we make $\log \Delta$ independent runs of $\mathcal{A}'$, the probability for obtaining a vertex cover in one of the runs is at least $1 - 1/\Delta$. Because the runs of $\mathcal{A}'$ are independent, they can be done in parallel. Let S' be the union of the $\log \Delta$ node sets computed by $\log \Delta$ runs of algorithm $\mathcal{A}$. The set S' can be computed in time $2t$. The size of S' is at most $2 \log \Delta$ times the size of an optimal vertex cover. With probability at least $1 - 1/\Delta$, S' is a vertex cover. In case that S' is not a vertex cover, we can turn S' into a vertex cover by adding all nodes which are adjacent to an uncovered edge. Let $\mathcal{OPT}$ be the size of an optimal vertex cover. The expected size of S' as described is

$$\mathrm{E}\left[|S'|\right] \;\leq\; \left(1 - \frac{1}{\Delta}\right) \cdot 2 \log \Delta \cdot \mathcal{OPT} + \frac{n}{\Delta} \;\leq\; \left(2 \log \Delta + 1 + \frac{1}{n-1}\right) \cdot \mathcal{OPT}$$

where n is the number of nodes of degree at least one. Note that because every node can cover at most Δ edges and since there are at least $n - 1$ edges, we have $(n-1)/\Delta \leq \mathcal{OPT}$. Because the lower bounds of Theorem 3.12 also hold for polylogarithmic approximation ratios, the MM part of the theorem follows.

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 a MIS on $L(G_k)$ and t' to compute a MM on G_k can only differ by a constant, $t \geq t' - \mathrm{O}(1)$. 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 squaring does not hurt and the theorem holds ($\log n = \Theta(\log n')$). □

Chapter 4

Distributed Graph Coloring

In Chapter 2, we have circumvented symmetry breaking by relaxing a given covering or packing problem to the corresponding linear program. This is of course only possible if a given combinatorial problem can be locally converted to an LP for which there is an efficient way to transform a fractional solution into an integer one. For the problems which are traditionally used to study symmetry breaking (e.g. MIS construction or graph coloring), the described LP approach does not work. One such problem is the focus of the present chapter in which we consider distributed algorithms for coloring the network graph.

4.1 The Minimum Graph Coloring Problem

A proper s-coloring of a graph $G = (V, E)$ is an assignment $\gamma : V \to [s]$ of colors between 1 and s to nodes such that adjacent nodes have different colors, that is, $(u, v) \in E \Rightarrow \gamma(u) \neq \gamma(v)$. Usually, the goal is to find a coloring of the nodes of a graph which uses a small number of colors, resulting in the minimum graph coloring problem. The minimum number of colors which are needed to properly color the vertices of a given graph G is called the chromatic number $\chi(G)$ of G.

Finding a small graph coloring is one of the most fundamental graph problems and has therefore been studied for a long time. Many network coordination primitives are based on colorings of the nodes of the network. For example, the assignment of frequencies or time slots in wireless networks are classical applications of minimum graph coloring [64, 119]. Like minimum dominating set, minimum graph coloring is one of the first problems which was shown to be NP-hard [53, 73]. A lot of progress has been made since then, showing that for general graphs, we cannot hope to find reasonably good colorings in polynomial time. In particular, unless P = NP, for every constant $\varepsilon > 0$, minimum graph coloring cannot be approximated better than $|V|^{1/7-\varepsilon}$ [20]. If NP $\neq$ ZPP, one can even show that minimum graph coloring cannot be approximated better than $\Omega(|V|^{1-\varepsilon})$ [43].

4.2 The Distributed Graph Coloring Problem

In the distributed setting, naturally the goal is to color the network graph G. Symmetries are usually broken either by randomness or by the use of node identifiers or other a priori node labelings. In this chapter, we concentrate on the second method since the main focus is on deterministic algorithms. We consider two variants of the distributed coloring problem. For the first variant, we make the common assumption that nodes have unique identifiers between 1 and some number $N \geq n$, where $n = |V|$. Often, it is even assumed that $N = n$. In the second more general variant, we assume that the algorithm starts with some proper coloring with colors from $\{1, \ldots, m\}$ of the graph. The goal then is to obtain a q-coloring with $q < m$. Clearly, the second variant of the problem is more general than the first variant because unique identifiers from 1 to N can be seen as an N-coloring of the graph. For the remainder of this chapter, the first variant is called the distributed coloring problem, whereas the second variant is called the distributed color reduction problem [35]. There are several reasons to look at the color reduction problem.

- Starting from an initial coloring is a natural extension of the classical problem where nodes need to have unique IDs. There might even be situations where it is reasonable to have locally unique IDs but impossible to guarantee globally unique IDs.
- It turns out that many things become clearer or simpler for the more general problem. In the next section for instance, we show that most topology information can be ignored.
- Most coloring algorithms in the literature work in the more general setting. They usually consist of several phases which reduce the colors from originally N to some final number.

In the previous section, we have summarized the most important results concerning the hardness of minimum graph coloring in a non-distributed setting. We can only hope to approximate coloring in any reasonable fashion in the unlikely case that ZPP = NP. In principle, the described hardness results cannot be adopted for the distributed models described in Section 1.2 since we allow arbitrary local computations. In fact as shown in Section 1.3.2, based on unique IDs, it is possible to compute an $O(\log n)$-approximation for minimum vertex coloring in randomized time $O(\log n)$ [93] or deterministic time $O(n^{O(1/\sqrt{\log n})})$ [9, 12, 108] in the $\mathcal{LOCAL}$ model by first computing a network decomposition. Nevertheless it seems unreasonable to study a distributed problem where exponential local computations are unavoidable. Therefore, when considering distributed coloring algorithms, we usually do not compare an obtained solution with a global optimal solution. We rather compare a computed coloring to what we can achieve by a simple sequential greedy algorithm. Therefore, for distributed algorithms, mostly, the ultimate goal is to achieve a $(\Delta+1)$-coloring or even just an $O(\Delta)$-coloring of the graph.

4.3 The Neighborhood Graph

For the analysis of deterministic algorithms, we use the concept of neighborhood graphs which was introduced in [91]. Neighborhood graphs are a means to formalize the possible views on which nodes have to base their decisions and the neighborhood relationships between different views. For a k-round algorithm, the node set of the neighborhood graph is the set of all possible views of a node after k communication rounds. Two nodes of the neighborhood graph are connected if and only if the corresponding two views can occur together at two adjacent nodes in the network graph G. In Section 1.2, we have seen that in the $\mathcal{LOCAL}$ model, each node v of a deterministic k-round algorithm computes an output value based on its view $\mathcal{V}_{v,k}$. Thereby, $\mathcal{V}_{v,k}$ is the labeled graph induced by the nodes in $\Gamma_k^+(v)$ excluding edges between nodes at distance exactly k from v. Throughout Section 4.3, we assume that coloring algorithms start with an a priori coloring instead of unique identifiers, that is, we consider the color reduction problem. Hence, the labels of nodes in $\mathcal{V}_{v,k}$ are colors between 1 and m.

To get some intuition for the neighborhood graph concept, we start with the simplest case where nodes only communicate for one round before choosing a color. The view $\mathcal{V}_{v,1}$ of a node v in a one-round algorithm consists of v and its neighbors. For coloring algorithms which start with an m-coloring, the view is a pair $(x_v, \Gamma_x(v))$ where x_v is the initial color of v and where $\Gamma_x(v)$ is the multi-set of the colors of the neighbors of v. The value of x_v and all elements of $\Gamma_x(v)$ are integers between 1 and m. If we only consider graphs with degrees at most Δ, we have $|\Gamma_x(v)| \leq \Delta$ and because we start with a valid coloring, we also have $x_v \notin \Gamma_x(v)$. Two views $(x_u, \Gamma_x(u))$ and $(x_v, \Gamma_x(u))$ are connected by an edge in the neighborhood graph if and only if $x_u \in \Gamma_x(v)$ and $x_v \in \Gamma_x(u)$. It is clear that two views can only be adjacent if these conditions holds. It is also not hard to see that all pairs of views which fulfill the given conditions can actually occur together in a graph with maximum degree Δ. Because adjacent nodes must not choose the same color, if two nodes u and v have views which are adjacent in the neighborhood graph, u and v must choose different colors. Assigning a color to each view $(x_v, \Gamma_x(v))$ therefore corresponds to finding a proper coloring of the neighborhood graph. Therefore, the chromatic number of the neighborhood graph characterizes the number of colors of a coloring which can be achieved by a one-round algorithm. Definition 4.1 generalizes the neighborhood graph concept for k-round coloring reduction algorithms applied to m-colored graphs of maximum degree Δ. The subsequent Lemma 4.1 then formally proves that the above observation about the chromatic number of the one-round neighborhood graph also holds for the general Definition 4.1.

Definition 4.1. ***(Neighborhood Graph)*** *The neighborhood graph* $\mathcal{N}_k(m, \Delta) = (\mathcal{V}_k(m, \Delta), \mathcal{E}_k(m, \Delta))$ *is defined as follows.*

- $\mathcal{V}_k(m, \Delta)$ *is the set of all properly* m*-colored, rooted* Δ*-regular trees of depth* k*.*

- *For a tree α in $\mathcal{V}_k(m,\Delta)$, let v_0^α be the root node of α and $v_1^\alpha,\dots,v_\Delta^\alpha$ be the neighbors of v_0^α. Further, for $i=0,\dots,\Delta$, T_i^α is the Δ-regular tree of depth $k-1$, rooted at v_i^α. Two trees α and β of the neighborhood graph are neighbors (i.e., $(\alpha,\beta)\in\mathcal{E}_k(m,\Delta)$) if and only if there are $i,j\in[\Delta]$ for which $T_0^\alpha = T_i^\beta$ and $T_0^\beta = T_j^\alpha$.*

Lemma 4.1. *Let $\mathcal{G}$ denote the class of m-colored graphs with maximum degree Δ. In k rounds, a deterministic distributed algorithm can color a graph of $\mathcal{G}$ with exactly $q=\chi(\mathcal{N}_k(m,\Delta))$ colors, that is, there is an algorithm which colors every graph of $\mathcal{G}$ with at most q colors and for every algorithm, there is a graph in $\mathcal{G}$ for which the resulting number of colors is at least q.*

Proof. We first prove that there is an algorithm which colors every properly m-colored graph G with q colors in k rounds. Slightly more general, we show that every t-coloring $\gamma_t : \mathcal{V}_k(m,\Delta)\to[t]$ of $\mathcal{N}_k(m,\Delta)$ can be turned into a k-round algorithm $\mathcal{A}_k(\gamma_t)$ for t-coloring arbitrary m-colored graphs with maximum degree Δ. Assume that we have a t-coloring γ_t which is globally known, that is, all nodes of G know γ_t. Note that this is no problem since the neighborhood graph and therefore also γ_t are independent of a particular network graph G. In a first phase of the algorithm $\mathcal{A}_k(\gamma_t)$, all nodes of G collect their complete neighborhoods up to distance k in k communication rounds. From this information, a node v of G constructs a tree α_v of depth k of its view as follows. The root of α_v is v and the children of the root node are the neighbors of v in G. The children of every other node u at depth at most $k-1$ of the tree α_v are all neighbors of u in G except of the parent node of u in α_v. Note that if v is part of a cycle of length smaller than $2k+1$ in G, there are nodes which occur at different places in v's tree. We m-color the tree by assigning each node the color of the corresponding node in G. Remember that we start our algorithm with an m-coloring of G. To obtain a Δ-regular tree of depth k, we add all the necessary additional nodes to the tree. In a root-to-leaf fashion, we assign the smallest possible color to each of the additional nodes. Note that each additional subtree is colored with the colors 1 and 2. The assignment of v's new color by algorithm $\mathcal{A}_k(\gamma_t)$ is now straightforward by applying γ_t, that is, v chooses the color $\gamma_t(\alpha_v)$. By the construction of α_v and by Definition 4.1, the trees $\alpha_u,\alpha_v\in\mathcal{V}_k(m,\Delta)$ of two neighboring nodes u and v in G are neighbors in $\mathcal{N}_k(m,\Delta)$. We therefore have $\gamma_t(\alpha_u)\neq\gamma_t(\alpha_v)$ which implies that $\mathcal{A}_k(\gamma_t)$ computes a valid t-coloring of G in k rounds.

To show that q also is a lower bound on the number of colors which a k-round algorithm can achieve, we have to prove that any two adjacent nodes $(\alpha,\beta)\in\mathcal{E}_k(m,\Delta)$ of $\mathcal{N}_k(m,\Delta)$ can occur as the k-neighborhoods of two adjacent nodes in an m-colored graph G with maximum degree Δ. However, $\mathcal{N}_k(m,\Delta)$ is defined such that $\mathcal{E}_k(m,\Delta)$ contains an edge (α,β) if and only if α and β can be the views of adjacent nodes on m-colored Δ-regular trees. Hence, the lemma is tight for Δ-regular trees of depth at least $k+1$. □

4.3.1 Properties of the Neighborhood Graph

As a consequence of Lemma 4.1, understanding distributed color reduction algorithms is tightly coupled to understanding the chromatic number of the neighborhood graph $\mathcal{N}_k(m,\Delta)$. Before starting a more detailed discussion, let us look at a few simple properties of $\chi\big(\mathcal{N}_k(m,\Delta)\big)$. From the definition of $\mathcal{N}_k(m,\Delta)$, it is immediate that

$$\chi\big(\mathcal{N}_k(m,\Delta)\big) \leq \chi\big(\mathcal{N}_{k-1}(m,\Delta)\big), \tag{4.1}$$

$$\chi\big(\mathcal{N}_k(m,\Delta)\big) \geq \chi\big(\mathcal{N}_k(m-1,\Delta)\big), \text{ and} \tag{4.2}$$

$$\chi\big(\mathcal{N}_k(m,\Delta)\big) \geq \chi\big(\mathcal{N}_k(m,\Delta-1)\big). \tag{4.3}$$

Lemma 4.2. *For every k and every $m \geq \Delta+1$, the chromatic number of the neighborhood graph $\mathcal{N}_k(m,\Delta)$ is*

$$\chi\big(\mathcal{N}_k(m,\Delta)\big) \geq \Delta+1.$$

Proof. Let $T_{k,i}$ be the Δ-regular tree of depth k, representing the distance k view of a node of color i in a $(\Delta+1)$-colored complete graph $K_{\Delta+1}$. By definition, the trees $T_{k,1},\ldots,T_{k,\Delta+1} \in \mathcal{V}_k(\Delta+1,\Delta)$ form a $(\Delta+1)$-clique in $\mathcal{N}_k(\Delta+1,\Delta)$ and hence $\chi(\mathcal{N}_k(\Delta+1,\Delta)) \geq \Delta+1$. The claim now follows by Equation (4.2). □

Corollary 4.3. *For every k and every $m \geq \Delta+1$, there are m-colored trees (graphs with chromatic number 2) which cannot be colored with less than $\Delta+1$ colors by a k-round color reduction algorithm.*

Lemma 4.2 and Corollary 4.3 give another reason why the goal of distributed coloring algorithms is to find a $(\Delta+1)$ or an $O(\Delta)$-coloring rather than to find a good approximation for the minimum coloring problem. In some way, Corollary 4.3 can also be seen as an extension to Theorem 3.1 in [91]. There, it is shown that Δ-regular trees of depth r cannot be colored with less than $\sqrt{\Delta}$ colors in less than $2r/3$ rounds. In [91], it is assumed that the nodes of the given Δ-regular tree have IDs from 1 to n. Corollary 4.3 extends the result to color reduction algorithms.

4.4 Deterministic One-Round Algorithms

After looking at a few general properties of $\mathcal{N}_k(m,\Delta)$ in Section 4.3.1, we make a detailed analysis of the most simple case where $k=1$ in this section. After one communication round, each node knows its own color and the colors of its at most Δ neighbors. Based on this information, all nodes have to decide on a new color.

Beyond being the simplest non-trivial case for general graphs, there are several reasons why looking at the one-round case is interesting. Most known distributed coloring algorithms are iterative applications of one-round color reduction algorithms [33, 55, 91]. From a practical point of view, this technique

is especially interesting because it results in algorithms for the $\mathcal{CONGEST}_{\mathrm{BC}}$ model. In every round, each node sends a message (its own color) to all of its neighbors. If we make the reasonable assumption that the number of colors m is polynomial in the number of nodes n of the network graph, the sizes of all sent messages are logarithmic in n. Understanding one-round color reduction algorithms is therefore essential to understanding an important class of distributed coloring algorithms.

The best known bounds for one-round coloring algorithms come from [91] where it is proven that

$$\chi\big(\mathcal{N}_1(m,\Delta)\big) \in \mathrm{O}\big(\Delta^2 \log m\big) \quad \text{and} \quad \chi\big(\mathcal{N}_1(m,2)\big) \in \Omega(\log\log m).$$

The upper bound is based on the existence of the following set system. For $t \in \mathrm{O}(\Delta^2 \log m)$, there are m subsets $S_1,\ldots,S_m$ of $\{1,\ldots,t\}$ such that for every $(\Delta+1)$-tuple of sets $S_{i_0},\ldots,S_{i_\Delta}$ for $i_j \in \{1,\ldots,m\}$, we have

$$S_{i_0} \setminus \bigcup_{j=1}^{\Delta} S_{i_j} \neq \emptyset.$$

A node v of color i_0 with neighbors of colors $i_1,\ldots,i_\Delta$ then chooses a number which is in S_{i_0} but not in S_{i_j} for $j \in \{1,\ldots,\Delta\}$ as its new color. Based on a result in [40], it is also proven in [91] that with such a set system, we cannot obtain a coloring with less than $\Omega(\Delta^2)$ colors. In the following, we extend this result to general one-round algorithms proving that for large enough m,

$$\chi\big(\mathcal{N}_1(m,\Delta)\big) \in \Omega\left(\frac{\Delta^2}{\log^2\Delta}\right).$$

4.4.1 Independent Sets of the Neighborhood Graph

We start our analysis of the chromatic number $\chi\big(\mathcal{N}_1(m,\Delta)\big)$ by looking at the structure of independent sets of the neighborhood graph $\mathcal{N}_1(m,\Delta)$. Because in a proper coloring of a graph, the nodes of each individual color form an independent set, finding a minimum coloring can equivalently be formulated as finding a minimum number of independent sets such that each node belongs to at least one independent set.

By Definition 4.1, the nodes $\alpha \in \mathcal{V}_1(m,\Delta)$ of the $(k=1)$-neighborhood graph are m-colored star graphs consisting of a center node and Δ peripheral nodes. Two stars α and β are neighbors in $\mathcal{N}_1(m,\Delta)$ if the center color of α occurs among the peripheral colors of β and vice versa. Figure 4.1 illustrates this by showing a part of $\mathcal{N}_1(6,3)$.

In general, for a star representing a node of $\mathcal{N}_1(m,\Delta)$, more than one peripheral node can have the same color. The following lemma shows that as long as we are only interested in the chromatic number of $\mathcal{N}_1(m,\Delta)$, we can w.l.o.g. assume that the peripheral nodes of a star in $\mathcal{V}_1(m,\Delta)$ all have distinct colors.

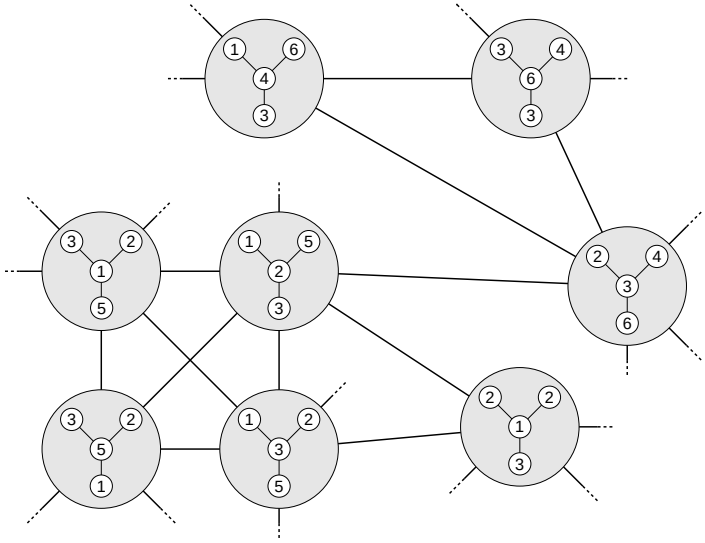

Figure 4.1: Looking into the one-round neighborhood graph $\mathcal{N}_1(6,3)$ of 6-colored degree 3 graphs.

Lemma 4.4. *For $m > \Delta$, let $\mathcal{N}'_1(m,\Delta)$ be the subgraph of $\mathcal{N}_1(m,\Delta)$ induced by all nodes in $\mathcal{V}_1(m,\Delta)$ for which the Δ peripheral nodes of the corresponding star graph have Δ different colors. We have*

$$\chi\big(\mathcal{N}'_1(m,\Delta)\big) \;=\; \chi\big(\mathcal{N}_1(m,\Delta)\big).$$

Proof. Because $\mathcal{N}'_1(m,\Delta)$ is a subgraph of $\mathcal{N}_1(m,\Delta)$, it is clear that

$$\chi\big(\mathcal{N}'_1(m,\Delta)\big) \;\leq\; \chi\big(\mathcal{N}_1(m,\Delta)\big).$$

To prove equality, we show that every t-coloring of $\mathcal{N}'_1(m,\Delta)$ can be extended to a t-coloring of $\mathcal{N}_1(m,\Delta)$. Let α be an m-colored Δ-star graph with at least two peripheral nodes of the same color. Hence, α is a node of $\mathcal{N}_1(m,\Delta)$ but not a node of $\mathcal{N}'_1(m,\Delta)$. Further, let β be a node of $\mathcal{N}'_1(m,\Delta)$ for which the set of peripheral colors is a super-set of the set of peripheral colors of α. By the definition of $\mathcal{N}_1(m,\Delta)$, the neighbor set of α is a sub-set of the neighbor set of β: $\Gamma(\alpha) \subset \Gamma(\beta)$. We can therefore safely color α with the color of β to obtain a t-coloring of $\mathcal{N}_1(m,\Delta)$. □

Note that Lemma 4.4 implies that for the one-round case, there is no difference between deterministic distributed coloring and deterministic distributed color reduction. As we will see in Section 4.5, this is not true for randomized one-round algorithms. Based on Lemma 4.4, we assume that $\mathcal{N}_1(m,\Delta)$ only

consists of m-colored stars with distinct peripheral colors for the remainder of Chapter 4. We can therefore represent nodes of $\mathcal{N}_1(m, \Delta)$ by a pair (x, Γ_x) where $x \in [m]$ is the center color and $\Gamma_x \subset 2^{[m]}$ is the set of peripheral colors. We have $x \notin \Gamma_x$ and $|\Gamma_x| = \Delta$. Two nodes (x, Γ_x) and (y, Γ_y) are adjacent in $\mathcal{N}_1(m, \Delta)$ if $x \in \Gamma_y$ and $y \in \Gamma_x$. The number of nodes N of $\mathcal{N}_1(m, \Delta)$ is

$$N \;=\; (\Delta + 1) \cdot \binom{m}{\Delta + 1}$$

because for each $(\Delta + 1)$-set of colors, there are $\Delta + 1$ nodes in $\mathcal{N}_1(m, \Delta)$ (one for each of the $\Delta + 1$ possible center colors).

The above observations allow to define a relation $\lhd_S$ among the colors $1, \ldots, m$ for each independent set S of $\mathcal{N}_1(m, \Delta)$. For an independent set S and two colors $x, y \in [m]$, we define $x \lhd_S y$ if and only if there is a node $(x, \Gamma_x) \in S$ for which $y \in \Gamma_x$. By the definition of $\mathcal{N}_1(m, \Delta)$ and because S is an independent set, $\lhd_S$ is antisymmetric:

$$x \lhd_S y \implies \neg(y \lhd_S x).$$

For convenience, we define the complementary relation as $x \not\lhd_S y := \neg(x \lhd_S y)$. In the described manner, it is not only possible to define an antisymmetric relation $\lhd_S$ for each independent set S, we can also find an independent set $S_\lhd$ for each antisymmetric relation $\lhd$. The independent set $S_\lhd$ consists of all nodes (x, Γ_x) for which $x \lhd y$ for all $y \in \Gamma_x$. We call $\lhd$ maximal if $\neg(y \lhd x) \Rightarrow x \lhd y$. The set $S_\lhd$ is a maximal independent set if and only if $\lhd$ is maximal. Conversely, every maximal independent set S has a maximal antisymmetric relation $\lhd_S$, that is, every maximal independent set can be defined by a maximal $\lhd$ as described. The following lemma shows that S is a maximum independent set whenever $\lhd$ is a total order on $[m]$.

Lemma 4.5. *Let N be the number of nodes of $\mathcal{N}_1(m, \Delta)$ and let $\prec$ be a total order on $[m]$. The resulting independent set $S_\prec$ is a maximum independent set of size $N/(\Delta + 1)$ of $\mathcal{N}_1(m, \Delta)$.*

Proof. Let D be a $(\Delta + 1)$-subset of $[m]$. For each $x \in D$, there is a node $(x, D \setminus \{x\})$ in $\mathcal{N}_1(m, \Delta)$. For a given total order $\prec$ on $[m]$, exactly one of the $\Delta + 1$ colors is the smallest w.r.t. $\prec$. Hence, exactly one of the $\Delta + 1$ nodes with colors in D is in $S_\prec$. Consequently, a $1/(\Delta + 1)$-fraction of all nodes of $\mathcal{N}_1(m, \Delta)$ is in $S_\prec$. Because every node of $\mathcal{N}_1(m, \Delta)$ is in a $(\Delta+1)$-clique (proof of Lemma 4.2), $S_\prec$ is a maximum independent set. □

In fact, it can even be shown that the maximum independent sets defined by total orders in the described way are the only maximum independent sets of $\mathcal{N}_1(m, \Delta)$. In other words, the relation $\lhd_S$ of a maximum independent set S is a total order on $[m]$. A direct implication of the structure of maximum independent sets of $\mathcal{N}_1(m, \Delta)$ is given by the following corollary. The fractional chromatic number of a graph G is defined as the size of the smallest fractional covering of the nodes of G with independent sets of G.

Corollary 4.6. *For all $k \geq 1$ and $m > \Delta$, the fractional chromatic number of the neighborhood graph is*

$$\chi_f\big(\mathcal{N}_k(m,\Delta)\big) \;=\; \Delta + 1.$$

Proof. By Lemma 4.5 and by symmetry, every node of $\mathcal{N}_1(m,\Delta)$ is in $m!/(\Delta+1)$ independent sets of size $N/(\Delta+1)$ defined by the $m!$ possible total orders on $[m]$. To obtain a fractional covering of the nodes with independent sets, we have to assign a weight x_S to each independent set S of G such that for each node v of G, the sum of the weights of all independent sets containing v is at least 1. By assigning $x_S = (\Delta+1)/m!$ to each of the $m!$ maximum independent sets of $\mathcal{N}_1(m,\Delta)$, we obtain a fractional covering of size $\Delta+1$. For $k \geq 1$, we therefore have

$$\chi_f\big(\mathcal{N}_k(m,\Delta)\big) \;\leq\; \chi_f\big(\mathcal{N}_1(m,\Delta)\big) \;\leq\; \Delta + 1.$$

In the proof of Lemma 4.2, we have shown that for all k and for all $m > \Delta$, $\mathcal{N}_k(m,\Delta)$ contains a $(\Delta+1)$-clique. Hence, $\chi_f(\mathcal{N}_k(m,\Delta)) \geq \Delta + 1$. □

Theorem 4.7. *For all $m > \Delta$, the chromatic number of the one-round neighborhood graph is*

$$\chi\big(\mathrm{N}_1(m,\Delta)\big) \;\leq\; (\Delta+1)^2(\ln m + 1).$$

Proof. The chromatic number $\chi(G)$ of a graph G equals the number of independent sets needed to cover all nodes of G. Hence, $\chi(G)$ is the solution of a minimum set cover instance. Because the integrality gap of minimum set cover is at most $\ln s + 1$, where s is the size of the largest set, we have

$$\chi(G) \;\leq\; \big(\ln(\alpha(G)) + 1\big)\chi_f(G)$$

where $\alpha(G)$ is the size of a maximum independent set of G. The theorem follows because

$$\alpha\big(\mathcal{N}_1(m,\Delta)\big) \;=\; \binom{m}{\Delta+1} \;<\; m^{\Delta+1}.$$

□

Remark: In [91], it has also been shown that $\chi\big(\mathcal{N}_1(m,\Delta)\big) \in \mathrm{O}(\Delta^2 \log m)$. Theorem 4.7 is a small constant improvement over the result of [91]. Up to lower-order terms, it is better by a factor of 4.

The upper bound given by Theorem 4.7 is strong if $\Delta^2 \ll m$. The following theorem shows that also for $m \leq \Delta^2$, the number of colors can be reduced in one round.

Theorem 4.8. *For all m, the chromatic number of the one-round neighborhood graph is at most*

$$\chi\big(\mathcal{N}_1(m,\Delta)\big) \;\leq\; \left\lceil m \cdot \frac{\Delta+1}{\Delta+2} \right\rceil \;=\; \left\lceil m \cdot \left(1 - \frac{1}{\Delta+2}\right) \right\rceil.$$

Proof. Let G be an m-colored graph of maximum degree Δ. We can construct a q-coloring for G for any q satisfying

$$q + \frac{q}{\Delta+1} \;=\; q \cdot \frac{\Delta+2}{\Delta+1} \;\geq\; m$$

as follows. Every node v with color $x_v \leq q$ keeps its color. We now still have to assign a color from $\{1,\ldots,q\}$ to all nodes having a color greater than q. Let the number of such colors be $t = m - q$. From the above condition, we have $t \leq \lfloor q/(\Delta+1) \rfloor$. We number those colors from x_0 to x_{t-1}, that is, we can for example set $x_i = m - i$. A node v with color x_i chooses a color from the set $\{i(\Delta+1)+1,\ldots,(i+1)(\Delta+1)\}$ which is not equal to the original color of any of v's neighbors. Because v can choose among $\Delta+1$ colors, such a color exists. Because nodes having colors x_i and x_j for $i \neq j$ choose their colors from disjoint color ranges, the obtained q-coloring is valid. The given upper bound on $\chi\big(\mathcal{N}_1(m,\Delta)\big)$ satisfies the condition for q because

$$\left\lceil m \cdot \frac{\Delta+1}{\Delta+2} \right\rceil \cdot \frac{\Delta+2}{\Delta+1} \;\geq\; m \cdot \frac{\Delta+1}{\Delta+2} \cdot \frac{\Delta+2}{\Delta+1} \;=\; m.$$

□

Theorem 4.8 gives a simple algorithm to transform an m coloring into a $\Delta+1$-coloring in $\mathrm{O}(\Delta \log(m/\Delta))$ rounds. In particular, the theorem shows that $\chi\big(\mathcal{N}_1(m,\Delta)\big) \leq m-1$ as long as $m \geq \Delta+2$. Combined with the $\mathrm{O}(\log^* m)$-time, $\mathrm{O}(\Delta^2)$-coloring algorithm, Theorem 4.8 implies an $\mathrm{O}(\Delta \log \Delta + \log^* n)$-time algorithm for coloring a graph with $(\Delta+1)$ colors. Because an MIS can be computed from a t-coloring in $\mathrm{O}(t)$ time, this also gives an algorithm for constructing an MIS in $\mathrm{O}(\Delta \log \Delta + \log^* n)$ time.

4.4.2 Lower Bound for One-Round Algorithms

In [91], it is shown that even for degree 2 graphs, every one-round coloring algorithm needs at least $\Omega(\log\log m)$ colors. Together with Lemma 4.2 this gives the following lower bound on the chromatic number of the one-round neighborhood graph:

$$\chi\big(\mathcal{N}_1(m,\Delta)\big) \;\in\; \Omega(\Delta + \log\log m).$$

In this section, we significantly improve this lower bound showing that the $\Omega(\Delta^2)$ lower bound for algorithms based on the technique described in [91] is almost tight for general one-round algorithms.

We start with an outline of the lower bound proof. Let $S_1,\ldots,S_q$ be q independent sets of $\mathcal{N}_1(m,\Delta)$ for some given m and Δ. If every node of $\mathcal{N}_1(m,\Delta)$ is in at least one of the q independent sets, the chromatic number of $\mathcal{N}_1(m,\Delta)$ is at most q. To prove a lower bound, the goal therefore is to show that if q is small

enough, we can find at least one node which is not in any of the independent sets. To argue about the role of a color x in independent set S, we define

$$\Phi_S(x) := \big\{y \,|\, x \not\lhd_S y\big\} \quad \text{and} \quad \varphi_S(x) := \big|\Phi_S(x)\big|.$$

Hence, $\Phi_S(x)$ denotes the set of colors y for which $y \notin \Gamma_x$ for all nodes $(x, \Gamma_x) \in S$. Consequently $\varphi_S(x)$ is the number of colors which do not occur in Γ_x for $(x, \Gamma_x) \in S$. Our goal is to show that for small enough q, we can find $\Delta + 1$ colors $x, y_1, \ldots, y_\Delta \in [m]$ such that

$$\forall i \in [q],\ \exists j \in [\Delta]\ :\ y_j \in \Phi_{S_i}(x) \tag{4.4}$$

for any possible choice of independent sets $S_1, \ldots, S_q$. If we can find such colors, it follows that $(x, \{y_1, \ldots, y_\Delta\})$ is not in any of the q independent sets $S_1, \ldots, S_q$. Hence, those q independent sets do not define a valid coloring of $\mathcal{N}_1(m, \Delta)$. For a given color x, the problem to find y_j for which Condition (4.4) is satisfied can be interpreted as an instance of minimum set cover. The colors $[m] \setminus \{x\}$ define the sets and the independent sets S_i define the elements. An independent set S_i is covered by a color y_j if $y_j \in \Phi_{S_i}(x)$. To prove that a one-round coloring algorithm needs more than q colors, we show that there is an $x \in [m]$ for which the optimal solution of the described set cover problem is at most Δ. In order to find such an x, we need the following lemma.

Lemma 4.9. *Let $A \subseteq [m]$ be a set of colors. For every independent set S of $\mathcal{N}_1(m, \Delta)$, we have*

$$\sum_{x \in A} \varphi_S(x) \ \geq\ \binom{|A|}{2}.$$

Proof. Because $\lhd_S$ is an antisymmetric relation, for any two colors $x_1, x_2 \in A$, we have

$$(x_1 \not\lhd_S x_2)\ \vee\ (x_2 \not\lhd_S x_1). \tag{4.5}$$

By definition, $\varphi_S(x)$ is the number of colors y for which $x \not\lhd_S y$. Hence, Equation (4.5) implies that for any two colors $x_1, x_2 \in A$, either $\varphi_S(x_1)$ or $\varphi_S(x_2)$ is increased by 1. Because the number of pairs in A is $\binom{|A|}{2}$, the claim follows. □

Lemma 4.9 implies that for each independent set S and each set A of colors, for at least half of the $|A|$ colors, $\varphi_S(x) \in \Omega(|A|)$. Hence, for each independent set S, many colors x are bad center colors because there are many colors which cannot occur as peripheral colors. In the following, we show that for small enough q, there must be a color x for which $\varphi_{S_i}(x)$ is large for almost all independent sets S_i, $i \in [q]$. To do so, for every x, we sort the values $\varphi_{S_i}(x)$ $(i = 1, \ldots, q)$ in increasing order. For all $j \in [q]$, we denote the j^{th} value $\varphi_{S_i}(x)$ in this sorted order by $h_j(x)$. Ties are broken arbitrarily. The next lemma shows that there is an x for which $h_j(x)$ grows linearly with j.

Lemma 4.10. *If* $t\big((m-q)t-q\big) > 2q(m-1)$*, there is an* $x \in [m]$ *for which*

$$\sum_{i=1}^{t} h_i(x) \;\geq\; m \quad \textit{and} \quad h_1(x) > 0.$$

Proof. Let $Q \subset [m]$ be the set of colors x for which there is an $i \in [q]$ such that $\varphi_{S_i}(x) = 0$, that is, $Q = \{x \in [m] \big| h_1(x) = 0\}$. For each independent set S, $\varphi_S(x) = 0$ for at most one color x. If $\varphi_S(x) = \varphi_S(y) = 0$ for two different colors $x \neq y$, this would imply that $x \lhd_S y \wedge y \lhd_S x$ which is not possible because $\lhd_S$ is antisymmetric. We therefore have $|Q| \leq q$. Let P be the complementary color set of Q, that is, $P = [m] \setminus Q$. We want to show that if $t\big((m-q)t-q\big) > 2q(m-1)$, there is an $x \in P$ for which $\sum_{i=1}^{t} h_i(x) \geq m$. For the sake of contradiction, assume that this is not the case and thus

$$\forall x \in P : \sum_{i=1}^{t} h_i(x) \leq m-1 \quad\Longrightarrow\quad \sum_{x\in P}\sum_{i=1}^{t} h_i(x) \leq |P|(m-1). \tag{4.6}$$

Let us take a closer look at the double sum in the right inequality of (4.6). The sum is over $|P| \cdot t$ different $h_i(x)$ values and therefore also over $|P| \cdot t$ different $\varphi_S(x)$ values. Let us denote the number of $\varphi_S(x)$ values for independent set S in the double sum of Inequality (4.6) by $a(S)$. By Lemma 4.9, we have

$$\sum_{x\in|P|}\sum_{i=1}^{t} h_i(x) \;\geq\; \sum_{j=1}^{q} \binom{a(S_j)}{2}, \quad \text{where} \quad \sum_{j=1}^{q} a(S_j) = |P| \cdot t. \tag{4.7}$$

For two integers A and B with $A < B$, we have

$$\begin{aligned}\binom{A}{2} + \binom{B}{2} \;&=\; \frac{A^2 - A + B^2 - B}{2} \;\geq\; \frac{A^2 - A + B^2 - B + 2(A-B+1)}{2} \\ &=\; \frac{A^2 + A - B^2 - 3B + 2}{2} \;=\; \binom{A+1}{2} + \binom{B-1}{2}.\end{aligned} \tag{4.8}$$

Combining Inequalities (4.8) and (4.7), we obtain

$$\sum_{x\in|P|}\sum_{i=1}^{t} h_i(x) \;\geq\; \sum_{j=1}^{q} \binom{a(S_j)}{2} \;\geq\; q \cdot \binom{|P|t/q}{2} \;=\; \frac{|P|t \cdot (|P|t-q)}{2q}.$$

Note that the above inequality also holds if $|P|t/q$ is not integral. Combined with Inequality (4.6), we therefore have

$$2q(m-1) \;\geq\; t(|P|t-q) \;\geq\; t\big((m-q)t-q\big)$$

which is a contradiction because we assumed that $t\big((m-q)t-q\big) > 2q(m-1)$. □

As described above, for each center color x, finding a node of $\mathcal{N}_1(m,\Delta)$ which is not covered by a given set of q independent sets of $\mathcal{N}_1(m,\Delta)$ can be seen as a minimum set cover problem where the colors $[m]$ are the sets and the q independent sets are the elements. Using Lemma 4.10, we can now prove that there is a color for which the above described set cover has a small fractional solution.

Lemma 4.11. *If $t\big((m-q)t-q\big) > 2q(m-1)$, there is a color $x \in [m]$ for which the above described minimum set cover problem has a fractional solution of size at most t.*

Proof. We have to assign a fractional value λ_y to each color $y \in [m] \setminus \{x\}$ such that

$$\forall i \in [q] : \sum_{y \in \Phi_{S_i}(x)} \lambda_y \;\geq\; 1. \tag{4.9}$$

We define

$$\lambda_y \;:=\; \frac{1}{\min\big\{\varphi_{S_i}(x) \,\big|\, y \in \Phi_{S_i}(x)\big\}}.$$

Clearly, this definition satisfies Condition (4.9). The value of the given fractional set cover solution is $\sum_y \lambda_y$. By the definition of $h_1(x)$, $h_1(x)$ colors y have a value $\lambda_y = 1/h_1(x)$. Further, at most $h_i(x)$ colors y have a value $\lambda_y = 1/h_i(x)$. Because of Lemma 4.10, there is a color x for which

$$\sum_y \lambda_y \;\leq\; \sum_{i=1}^{t} h_i(x)\frac{1}{h_i(x)} \;=\; t.$$

which concludes the proof. Note that we need that $h_1(x) > 0$ because the independent set S for which $\varphi_S(x) = 0$ contains all nodes (i.e., stars) with center color x. □

Based on Lemma 4.11, we can now also find a color x for which the described set cover problem has a small integer solution. This allows to derive the next lemma.

Lemma 4.12. *We have $\chi\big(\mathcal{N}_1(m,\Delta)\big) > q$ if the following condition holds:*

$$\Delta\left(\Delta - \frac{q\ln(eq)}{m-q}\right) \;>\; \frac{2(m-1)q\ln^2(eq)}{m-q}. \tag{4.10}$$

Proof. Let s be the size of the largest set of some given minimum set cover problem. By the analysis of the set cover greedy algorithm [31], the integrality gap of the minimum set cover problem is at most $H(s) \leq \ln s + 1 = \ln(es)$. The largest set of the set cover instance considered in Lemma 4.11 has size less than q. The integrality gap of the considered set cover problem therefore is bounded by $\ln(eq)$. We have seen that if there is some center color x for which there is a solution of size Δ for the problem of covering independent sets with colors, at

least one node of $\mathcal{N}_1(m,\Delta)$ is not covered by any of the independent sets. It therefore follows from Lemma 4.11 that $\chi\big(\mathcal{N}_1(m,\Delta)\big) > q$ if

$$\Delta \geq t\ln(eq) \quad \text{and} \quad t\big((m-q)t-q\big) > 2q(m-1). \tag{4.11}$$

Plugging the first inequality of (4.11) into the second inequality of (4.11), yields

$$\Delta\left(\Delta - \frac{q\ln(eq)}{m-q}\right) \;>\; \frac{2(m-1)q\ln^2(eq)}{m-q}.$$

□

This allows us to compute a lower bound for one-round coloring algorithms as given by the following theorem.

Theorem 4.13. *For $m \in \Omega(\Delta^2/\log\Delta)$, the chromatic number of the one-round neighborhood graph is*

$$\chi\big(\mathcal{N}_1(m,\Delta)\big) \;\in\; \Omega\left(\frac{\Delta^2}{\log^2\Delta} + \log\log m\right).$$

Proof. We have to show that Inequality (4.10) of Lemma 4.12 holds for $q \in \Omega(\Delta^2/\log^2\Delta)$ if $m \in \Omega(\Delta^2/\log\Delta)$. Hence, we assume that $m = d\Delta^2/\ln\Delta$ for some $d \in \Omega(1)$ and show how to find a $c \in \Omega(1)$ such that Inequality (4.10) holds for $q = c\Delta^2/\ln^2\Delta$. For $c < d$, we have

$$m - q \;=\; \frac{\Delta^2}{\ln\Delta}\cdot\left(d - \frac{c}{\ln\Delta}\right) \;\geq\; \frac{\Delta^2}{\ln\Delta}\cdot\frac{\ln\Delta - 1}{\ln\Delta}.$$

Hence, there is a small enough constant c for which the left-hand side of Inequality (4.10) is bounded by

$$\begin{aligned}\Delta\left(\Delta - \frac{q\ln(eq)}{m-q}\right) \;&\geq\; \Delta\left(\Delta - \frac{c\Delta^2(\ln(ec) + 2\ln\Delta - 2\ln\ln\Delta)\ln^2\Delta}{\ln^2\Delta\cdot\Delta^2(\ln\Delta - 1)}\right)\\ &>\; \Delta(\Delta-1).\end{aligned}$$

There is a constant c' such that

$$\frac{m}{m-q} \;\leq\; \frac{d\Delta^2\ln^2\Delta}{\ln\Delta\cdot\Delta^2(\ln\Delta-1)} \;\leq\; c'.$$

For the right-hand side of Inequality (4.10), we therefore obtain

$$\frac{2(m-1)q\ln^2(eq)}{m-q} \;\leq\; \frac{2c'c\Delta^2(\ln(ec) + 2\ln\Delta - 2\ln\ln\Delta)^2}{\ln^2\Delta} \;<\; \Delta(\Delta-1)$$

if the constant c is chosen small enough. The $\Omega(\log\log m)$ lower bound follows from the proof of Theorem 2.1 in [91]. □

4.4.3 Lower Bound for Iterative One-Round Color Reduction

We have seen that many of the published coloring algorithms are iterative applications of one-round color reduction schemes. It would therefore be nice to have a lower bound for one-round color reduction steps which allows to analyze the potential of such iterative color reduction algorithms. The lower bound given by Theorem 4.13 can only be applied if m is large enough. However, if we want to lower bound the number of one-round color reduction steps needed to for example achieve an $O(\Delta)$-coloring, we have to bound $\chi(\mathcal{N}_1(m,\Delta)$ for small m. Based on the condition given by Lemma 4.12, we obtain the following theorem.

Theorem 4.14. *Assume that $m = \rho\Delta$ for an arbitrary $\rho > 1$. We have*

$$\chi\big(\mathcal{N}_1(m,\Delta)\big) \;\geq\; \left(1 - \mathrm{O}\left(\frac{\rho \log^2 \Delta}{\Delta}\right)\right) \cdot m.$$

Proof. By Lemma 4.12, we have $\chi(\mathcal{N}_1(m,\Delta)) > q$ if

$$\Delta\left(\Delta - \frac{q\ln(eq)}{m-q}\right) \;>\; \frac{2(m-1)q\ln^2(eq)}{m-q}.$$

By substituting $\rho\Delta$ for m and $(1-\varepsilon)m$ for q, we obtain

$$\varepsilon\Delta\big(\Delta - (1-\varepsilon)\ln(e(1-\varepsilon)\rho\Delta)\big) \;>\; 2(\rho\Delta-1)(1-\varepsilon)\ln^2(e(1-\varepsilon)\rho\Delta) \quad (4.12)$$

W.l.o.g., we can assume that $\rho \in \mathrm{O}(\Delta/\log^2\Delta)$. We thus have $\ln^2(e(1-\varepsilon)\rho\Delta) \leq c\ln\Delta$ for some constant c. The left-hand side and the right-hand side of Inequality 4.12 can be bounded as follows:

$$\begin{aligned}\varepsilon\Delta\big(\Delta - (1-\varepsilon)\ln(e(1-\varepsilon)\rho\Delta)\big) &\geq \varepsilon(\Delta^2 - c\Delta\ln\Delta),\\ 2(\rho\Delta-1)(1-\varepsilon)\ln^2(e(1-\varepsilon)\rho\Delta) &\leq 2c\rho\Delta\ln^2\Delta.\end{aligned}$$

Inequality 4.12 therefore holds if

$$\varepsilon \;\geq\; \frac{2c\rho\Delta\ln^2\Delta}{\Delta^2 - c\Delta\ln\Delta} \;\in\; \mathrm{O}\left(\frac{\rho\log^2\Delta}{\Delta}\right)$$

which concludes the proof. □

Theorem 4.14 shows that if we start with a $\rho\Delta$-coloring, we need at least $\Omega\big(\Delta/(\rho\log^2\Delta)\big)$ rounds to reduce the number of colors by a constant factor. Combining with the $\Omega(\log^* m)$ lower bound for coloring a graph with $\mathrm{O}(\Delta)$ colors [91], we obtain the following bound on the number of one-round color reduction steps needed to color a given graph.

Corollary 4.15. *Assume that $m \geq c\beta(\Delta+1)$ for some constant $c > 1$ and some $\beta > 1$. The number of one-round color reduction steps needed to obtain a $\beta(\Delta+1)$-coloring is at least $\Omega\big(\Delta/(\beta\log^2\Delta)\big)$. In particular, any $\mathrm{O}(\Delta)$-coloring algorithm which is based on iterative applications of one-round color reduction steps, needs at least $\Omega(\Delta/\log^2\Delta + \log^* m)$ rounds.*

4.5 Randomized Distributed Coloring

Up to this point, we have focused on deterministic coloring algorithms. Let us now explore the potentials of randomized coloring algorithms. In [91], it has been proven that randomization does not help for distributed coloring algorithms. However, in the proof of [91], it is implicitly assumed that a k-round randomized coloring algorithm colors a graph with s colors if the algorithm always stops with a proper s-coloring after k rounds. The proof of [91] does not hold if it suffices that the number of rounds is k or the number of colors is s in expectation or with high probability. In the following, we show that on the one hand, randomization does not help for k-round color reduction because we possibly have $n \gg m, \Delta$. On the other hand, random distributed coloring algorithms are strictly stronger than deterministic ones if the results for either time complexity or number of colors only have to hold with high probability.

Theorem 4.16. *Assume that there is a randomized k-round color reduction algorithm $\mathcal{A}_\mathcal{R}$ which colors every m-colored graph G with maximum degree Δ with q colors with probability $p(m,\Delta) > 0$. Then there exists a deterministic k-round color reduction algorithm $\mathcal{A}_\mathcal{D}$ which colors G with q colors, that is,*

$$\chi\big(\mathcal{N}_k(m,\Delta)\big) \leq q.$$

Proof. The theorem follows from Theorem 5.1 in [105] where it is proven that randomization does not help to compute a locally checkable labeling if the number of rounds and the number of labels do not depend on the number of nodes n of the network graph. For completeness, we outline a possible proof.

For the sake of contradiction assume that the number of colors computed by the randomized color reduction algorithm $\mathcal{A}_\mathcal{R}$ is $q < \chi\big(\mathcal{N}_k(m,\Delta)\big)$. By Lemma 4.1, there is a graph G for which every deterministic color reduction algorithm needs at least $\chi\big(\mathcal{N}_k(m,\Delta)\big)$ colors. Hence, there is a positive probability $\varepsilon > 0$ that $\mathcal{A}_\mathcal{R}$ fails to compute a q-coloring for G. We construct an m-colored graph G' by taking t m-colored copies of G. Since the random choices of the nodes of different copies of G are independent, algorithm $\mathcal{A}_\mathcal{R}$ independently fails for each copy of G with probability ε. If we choose t large enough, the success probability $(1-\varepsilon)^t$ becomes smaller than $p(m,\Delta)$ which is contradiction to the assumption that $q < \chi\big(\mathcal{N}_k(m,\Delta)\big)$. □

The reason that randomization does not help for color reduction is that the number of nodes n of the network graph can be arbitrarily greater than m and Δ. If we allow the number of colors q to depend on n or if we assume that m is a function of n, Theorem 4.16 does not hold. In particular, if we assume that initially all nodes have unique IDs between 1 and N, that is, $m = N$, randomization might help. We show that for certain Δ, m, and n, randomized one-round coloring algorithms can beat the one-round lower bound for deterministic algorithms given by Theorem 4.13. For deterministic algorithms, we assumed that in one round, every node can learn the labels of all its neighbors. In a randomized algorithm, it is additionally possible to collect all random decisions (i.e., all

Algorithm 8 Randomized coloring in one round (code for node v)

1: choose color t_v uniformly at random from $\{1, \ldots, \lceil \Delta / \ln n \rceil\}$;
2: **send** ID(v) and t_v to all neighbors;
3: let G_{t_v} be the graph induced by all nodes u with $t_u = t_v$;
4: let Δ_{t_v} be the maximum degree of G_{t_v};
5: compute $O(\Delta_{t_v}^2 \log m)$-coloring of $G_{t_v} \Longrightarrow$ color y_v;
6: color $x_v := y_v \lceil \Delta / \ln n \rceil + t_v - 1$

random bits) of the neighbors. Algorithm 8 describes a randomized algorithm which colors a given network graph in one round. The following theorem shows that Algorithm 8 computes a small, proper coloring. We assume that all nodes know Δ, m, and n.

Theorem 4.17. *For any constant c and with probability $1 - 1/n^c$, Algorithm 8 computes a proper coloring of the network graph G with $O(\Delta \log n \log m)$ colors in one round. The choice of the constant c only influences the number of colors by a constant factor.*

Proof. Let Δ_T be $\max\{\Delta_t | t \in [\lceil \Delta / \ln n \rceil]\}$. We first show that Algorithm 8 computes a $O(\Delta \Delta_T^2 \log m / \log n)$-coloring in one round. The only places where something has to be computed are Lines 1, 5, and 6. The only problem occurs in Line 5. If v knew Δ_{t_v}, an $O(\Delta_{t_v}^2 \log m)$-coloring could be computed by using Theorem 4.7 or by applying the algorithm of [91]. If we are willing to pay a small constant factor in the number of colors, the described algorithms can be adapted to the case where the maximum degree is not known. Let $\Delta_i := 2^i$. Assume that we are given $O(\Delta_i^2 \log m)$-colorings of $\mathcal{N}_1(\Delta_i, m)$ for all i such that different colors are used for different Δ_i. A node v with degree $\delta(v)$ can choose its color according to the respective color of the neighborhood graph for the smallest $\Delta_i \geq \delta(v)$.

In order to complete the proof, it therefore suffices to show that $\Delta_T \in O(\log n)$ with probability $1 - 1/n^c$. To do so, we compute a high probability upper bound for the degree $\delta_{t_v}(v)$ of v in G_{t_v} using Chernoff (Theorem 1.3). Let $Q := \lceil \Delta / \ln n \rceil$. The probability that a neighbor u of v chooses the same color in Line 1 (i.e., $t_u = t_v$) is $1/Q$. The expected number of neighbors u of v for which $t_u = t_v$ is therefore at most $\Delta / Q \leq \ln n$. By Theorem 1.3, we thus get

$$\Pr[\delta_{t_v}(v) \geq \kappa e \ln n] \;<\; \left(\frac{e^{\kappa e - 1}}{(\kappa e)^{(\kappa e)}} \right)^{\ln n} \;=\; \frac{1}{n^{1 + \ln \kappa \cdot \kappa e}}.$$

We then have

$$\Pr[\Delta_T \geq \kappa e \ln n] \;\leq\; n \cdot \Pr[\delta_{t_v}(v) \geq \kappa e \ln n] \;<\; \frac{1}{n^{\ln \kappa \cdot \kappa e}}.$$

Choosing κ such that $c = \ln \kappa \cdot \kappa e$ completes the proof. □

If $\log n \ll \Delta$, Theorem 4.17 together with the lower bound of Theorem 4.13 shows that randomization can help in distributed coloring.

In Line 5, we apply a deterministic one-round algorithm which colors an m-colored graph with maximum degree Δ with $O(\Delta^2 \log m)$ colors. In [91], it is shown that by applying such an algorithm $O(\log^* m)$ times, it is possible to compute an $O(\Delta^2)$-coloring in $O(\log^* m)$ rounds. If we replace Line 5 of Algorithm 8 by this better (but slower) algorithm, we obtain the following corollary to the above theorem.

Corollary 4.18. *For any constant c, it is possible to color any m-colored graph G with maximum degree Δ with $O(\Delta \log n)$ colors in $O(\log^* m)$ rounds with probability $1 - 1/n^c$.*

The next corollary follows from the above corollary and from Theorem 4.8.

Corollary 4.19. *For any constant c, it is possible to color a graph with $O(\Delta+1)$ colors in $O(\Delta \log\log n)$ rounds with probability $1 - 1/n^c$.*

4.6 Time Division Multiple Access in Two Rounds

In Section 1.1.3, we have described time division multiple access (TDMA) schemes as a possible application of colorings of the network graph. The goal of a TDMA scheme is to assign time slots to nodes such that possibly interfering nodes use different time slots. The signals of two nodes u and v of $G = (V, E)$ can interfere if they both communicate with each other or if both communicate with the same additional node w. Hence, we have to assign different time slots to nodes which are at distance 1 or 2 from each other [64, 119].

Let G^2 be the graph with node set V and an edge between u and v if and only if $d_G(u, v) \leq 2$. Clearly, a coloring of graph G^2 yields a valid assignment of time slots. Each color represents a time slot. Since interfering nodes u and v have different colors, they also use different time slots. However, even if we are given an optimal coloring of G^2, the assignment of time slots might be far from optimal. Assume for instance that a large part of the graph G^2 can be colored using $t \ll \chi(G^2)$ colors whereas the $\chi(G^2)$ colors are only needed for a very small part of the graph. With the straightforward approach, each of the $\chi(G^2)$ colors is assigned to one of $\chi(G^2)$ time slots. This means that in the part of the graph where t colors would suffice, only roughly a $t/\chi(G^2)$-fraction of the time slots is really assigned. This results in a TDMA scheme for which a large fraction of the total bandwidth is unused. In reality, the problem becomes even worse since we cannot hope to find a coloring which is close to an optimal one. Because we need $\chi(G^2)$ time slots for the part of G^2 having a high chromatic number, the only solution is to assign several slots to nodes in parts of G^2 having a small chromatic number. In the following, we show that two communication rounds suffice to assign an $O(1/\delta_2(v))$-fraction of all slots to a node v with degree $\delta_2(v)$ in G^2. A node v only needs to know the IDs of

all its neighbors in G^2 and thus the 2-hop neighbors in G to be able to use an $O(1/\delta_2(v))$-fraction of all time slots. For convenience, we define $H := G^2$. The maximum degree of H is denoted by Δ_H and the degree of a node v in H is denoted by $\delta_H(v)$. When formulated for graph H, the problem is to assign a set of colors to each node v of H such that the color sets of adjacent nodes are disjoint and such that the number of colors per node is maximized. In order to obtain a two-round algorithm for G, the algorithm for H must terminate after a single round.

Theorem 4.20. *Let C be the total number of colors and assume that all nodes of a graph H with maximum degree Δ_H have a unique ID between 1 and N. For $C \in O(\Delta_H^2 \log N)$, there is a one-round algorithm which assigns colors to nodes such that a node v with degree $\delta_H(v)$ gets assigned $\Omega(\Delta_H^2 \log N/\delta_H(v))$ colors and such that the color sets assigned to adjacent nodes are disjoint.*

Proof. We have seen in Lemma 4.5 that each total order $\prec$ on $\{1,\dots,N\}$ induces a maximum independent set $S_\prec$ of $\mathcal{N}_1(N,\Delta_H)$. A node (x,Γ_x) of $\mathcal{N}_1(N,\Delta_H)$ is in $S_\prec$ if and only if $\forall y \in \Gamma_x : x \prec y$. If we assign a color x to all nodes in $S_\prec$, in the graph H, a node u can choose color x if $\mathrm{ID}(u) \prec \mathrm{ID}(v)$ for all neighbors v of u. We have seen in Theorem 4.7 that $(\Delta_H+1)^2(\ln N+1)$ such maximum independent sets suffice to cover each node of $\mathcal{N}_1(N,\Delta_H)$ at least once. How many independent sets are needed such that every node u of degree $\delta_H(u) \le \Delta_H$ is covered at least $\Omega(\Delta_H^2 \log N/\delta_H(u))$ times?

To analyze this, we have to extend the node set of $\mathcal{N}_1(N,\Delta_H)$ to all nodes (x,Γ_x) for which $x \notin \Gamma_x$ and for which $|\Gamma_x| \le \Delta_H$. Note that before, we could assume that $|\Gamma_x| = \Delta_H$. The number of nodes K of the extended neighborhood graph is

$$K = \sum_{i=1}^{\Delta_H} \binom{N}{i+1}(i+1) \le \sum_{i=1}^{\Delta_H} \left(\frac{eN}{i+1}\right)^{i+1} \cdot (i+1) \le \sum_{i=1}^{\Delta_H} (eN)^{i+1} \in O\big((eN)^{\Delta_H+1}\big).$$

We show that $O(\Delta_H^2 \log N)$ independent set are enough to cover all K nodes sufficiently often using the probabilistic method. Assume that we choose t independent sets at random. If the probability for not covering all K nodes sufficiently often is smaller than 1, there is a non-zero probability that the choice of independent sets is successful. This proves that it is possible to cover all K nodes (x,Γ_x) of the neighborhood graph sufficiently often ($\Omega(\Delta_H^2 \log N/|\Gamma_x|)$ times) with t independent sets. To choose a maximal independent set $S_\prec$ at random, we choose a total order $\prec$ uniformly at random. The probability that a given node (x,Γ_x) is covered by $S_\prec$ is $1/(|\Gamma_x|+1)$. Let X be the random variable denoting the number of times (x,Γ_x) is covered. If we choose $t = 8(\Delta_H+1)\ln K = O(\Delta_H^2 \log N)$, the probability that $X < t/(2(|\Gamma_x|+1))$

can be bounded using Chernoff (Theorem 1.2). We have

$$\begin{aligned}\Pr\left[X < \left(1-\frac{1}{2}\right)\frac{t}{|\Gamma_x|+1}\right] &< e^{-\frac{t}{8(|\Gamma_x|+1)}} < e^{-\frac{t}{8(\Delta_H+1)}} \\ &= e^{-\frac{8\ln K(\Delta_H+1)}{8(\Delta_H+1)}} = \frac{1}{K}.\end{aligned}$$

Therefore the probability that at least one of the K nodes (x,Γ_x) is not covered $t/(2(|\Gamma_x|+1))$ times is less than 1. This implies that there exist t independent sets of $\mathcal{N}_1(N,\Delta_H)$ such that all of the K nodes (x,Γ_x) are covered at least $t/(2(|\Gamma_x|+1))$ times. □

Remark: Note that the TDMA scheme given by Theorem 4.20 is not optimal. If v is the center of a $\Theta(\sqrt{n})$-regular tree of depth 2, the degree of v in G^2 is n and the degrees of all other nodes in G^2 are $\Theta(\sqrt{n})$. The described algorithm assigns only a $\Theta(1/n)$-fraction of all time slots to v although it would be possible to assign $\Theta(1/\sqrt{n})$-fractions to all nodes. However, the algorithm is asymptotically optimal for two rounds because it is not possible for v to distinguish between the given graph G^2 an a case where G^2 is a complete graph K_n in two rounds.

Chapter 5

On The Locality of Bounded Growth

5.1 Introduction

In Chapters 2–4, we have considered the distributed complexity of several basic graph theoretic problems. Although there is clearly room from improvements, for most of the problems the presented upper bounds are at least in the same range as the given lower bounds. We have for example seen that in a constant number of rounds the achievable approximation ratio for covering and packing problems is a constant root of n or Δ. What do these results mean for real networks? Given that we have to solve such a problem, can we really not do any better?

Up to here, we always assumed that the topology of the network can be an arbitrary graph. Especially when proving the lower bounds in Chapter 3, we made extensive use of this assumption. The constructed cluster tree is definitely far from any graph occurring in a real-world network. In the following, we will therefore drop this general graph assumption. We will look at a restricted class of graphs which we call *growth-bounded graphs*. We believe that the underlying assumptions for this graph class are realistic in many situations, especially when considering wireless ad hoc and sensor networks.

5.1.1 Unit Disk Graphs

Very often, ad hoc and sensor networks are modeled as *unit disk graphs (UDG)*. In a UDG, it is assumed that all network nodes are located on a two-dimensional plane. All wireless nodes have the same transmission range, two nodes can communicate directly with each other whenever they are within each other's transmission range. Hence, a graph G is a unit disk graph exactly if there is an assignment of coordinates in $\mathbb{R}^2$ such that there is an edge between two nodes if and only if their distance is at most 1.

Many of the classical graph theoretic problems of Chapters 2–4 are much easier to approximate on UDGs than on general graphs, at least in a non-distributed setting. A maximal independent set for example is a constant approximation for minimum dominating set and for maximum independent set and a $(\Delta + 1)$-coloring is a constant approximation for minimum graph coloring (see e.g. [98]). For the minimum vertex cover problem, things are even simpler. It can be shown that the set of all non-isolated nodes of a UDG is a constant approximation for the MVC problem. As a consequence, there is a trivial constant-time distributed approximation algorithm for MVC on UDGs. This is in sharp contrast to the distributed MVC lower bound of Chapter 3 for general graphs.

Although the UDG model is the most popular graph model for ad hoc and sensor networks, it appears clear that the underlying assumptions are much too ideal to be realistic. The devices of a wireless network are usually not on a plane surface with no obstacles in-between. Even if they were, we could not assume that all transmission radii are exactly equal. Nevertheless, some of the properties of UDGs are realistic. In the following, the most basic properties of UDGs are used to develop the notion of growth-bounded graphs.

5.2 Growth-Bounded Graphs

In Section 5.1.1, we have mentioned that on a UDG, MDS, maximum independent set, and graph coloring can be approximated by computing an MIS or a $(\Delta+1)$-coloring, respectively. What properties of the UDG do we need to prove these statements?

Let us look at the MDS problem. Assume that we are given a UDG G and an MIS S of G. From the definition of an MIS, it is clear that S is a dominating set of G. To prove that S is only by a small factor larger than an optimal dominating set, consider a single node v of G. Let $d(v)$ be the number of dominators in $\Gamma^+(v)$, that is, $d(v) = |S \cap \Gamma^+(v)|$. If $v \in S$, we have $d(v) = 1$ because S is an MIS. If $v \notin S$, we show that $d(v) \leq 5$. For the sake of contradiction, assume that $d(v) \geq 6$. In that case, six nodes of S are in a circle of radius 1 and center v. Because S is an independent set, the distance between any two of these six nodes has to be larger than one. When looking from the center v of the circle, the angle between any two points must be larger than $60°$. However because the whole disk has only $360°$, this is not possible and we therefore have $d(v) \leq 5$. Assigning $y_i = 1/5$ to each node $v_i \in S$ therefore gives a feasible solution for ($\mathrm{DP}_{\mathrm{DS}}$), the dual of the dominating set LP ($\mathrm{LP}_{\mathrm{DS}}$). By LP duality, this implies that an MIS is a 5-approximation for MDS on UDGs.

We only use the fact that G is a UDG in the above argumentation when bounding the number of dominators $d(v)$ among the neighbors of a node v. Thereby, we applied the fact the no node of a UDG can have more than 5 independent (i.e., pairwise non-adjacent) neighbors. Such a graph is called

$K_{1,6}$-free because there is no set of 7 nodes for which the induced subgraph is the complete bipartite graph $K_{1,6}$. In fact, $K_{1,6}$-freeness also suffices to prove the described statements about maximum independent set and minimum graph coloring.

Because many of the UDG complexity results are based on the $K_{1,6}$-freeness of UDGs, we would like to keep this property when modeling topologies of ad hoc or sensor networks. But is it realistic to assume that the graphs of ad hoc and sensor networks are $K_{1,t}$-free for some constant t? Although we cannot assume that nodes can directly communicate if and only if their distance is at most 1, assuming that close nodes can directly exchange messages rather than far-away nodes is definitely reasonable. It is therefore also reasonable to assume that as soon as there are enough nodes within some bounded area on the plane or some bounded volume in space, there are nodes which can communicate with each other. Because all neighbors of a node v have to be within a bounded area or volume to communicate with v via radio, this implies $K_{1,t}$-freeness for some bounded t. It even implies a stronger property which we use to define growth-bounded graphs.

Definition 5.1. ***(Growth-Bounded Graph)*** *We call a graph G f-growth-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.*

Note that f is a function of r and does not depend on any other property of G. In particular, $f(r)$ is independent of the number of nodes n or the largest degree Δ. If $f(r)$ is a polynomial in r, we say that G is *polynomially growth-bounded*. Note also that an f-growth-bounded graph is $K_{1,f(1)+1}$-free.

To get some intuition for the above definition, let us determine the function f for UDGs. All nodes in the r-neighborhood $\Gamma_r^+(v)$ of a node v are in a disk of radius r around v. Let us look at an independent set S of the subgraph of G induced by $\Gamma_r^+(v)$. Two nodes of S have distance more than 1. Therefore, the disk of radius $1/2$ around each node $u \in S$ contains no other node of S. The number of nodes in S therefore is bounded by the number of times the area of a disk of radius $1/2$ fits into the area of a disk of radius r and hence $f(r) \leq \pi r^2/(\pi/4) = 4r^2$. If we extend the UDG definition to arbitrary constant-dimensional Euclidean spaces, the growth function is $f(r) \in \mathrm{O}(r^d)$. Another simple example which satisfies Definition 5.1 are constant-degree graphs. If the degree of every node of a graph G is bounded by a constant $\Delta > 1$, we have $f(r) \in \mathrm{O}(\Delta^r)$, that is, the growth of G is exponentially bounded in this case.

5.2.1 Graph Classes Having Bounded Growth

In this section, we look at other graph models which imply bounded growth. The following definition introduces a natural extension of the UDG model where we assume that nodes 'live' in an arbitrary metric space instead of the Euclidean plane.

Definition 5.2. (Unit Ball Graph) *Let $M = (X, d)$ be some finite metric space with $n = |X|$ points and distance function $d : X \times X \to \mathbb{R}$. The graph $G = (V, E)$ with node set $V = X$ and edge set $E = \{(u, v) \in V \times V \,|\, d(u, v) \leq 1\}$ is called the* unit ball graph *induced by M.*

If the metric M is a finite sub-space of the 2-dimensional Euclidean plane, the unit ball graph (UBG) definition coincides with the UDG definition. In order to get something interesting, we have to restrict the metric space M in some way. Note that every undirected graph G is a UBG, namely the UBG induced by the shortest path metric of G.

In order to bound the growth of a metric, we use the *doubling dimension* as introduced in [59]. The doubling dimension of a metric space is the smallest ρ such that every ball can be covered by at most 2^ρ balls of half the radius. If ρ is a constant, we say that the given metric is *doubling*. In analogy, we call a UBG doubling if the underlying metric space is doubling. The following lemma shows that doubling UBGs are polynomially growth-bounded.

Lemma 5.1. *Let $G = (V, E)$ be a UBG which is induced by some metric $M = (X, d)$ with doubling dimension ρ. G is f-growth-bounded for $f(r) \in \mathrm{O}(r^\rho)$.*

Proof. Consider an independent set S of the subgraph of G induced by an r-neighborhood $\Gamma_r^+(v)$ for some node $v \in V$. For every $u \in \Gamma_r^+(v)$, the ball with radius $1/2$ around u contains at most one node of S because S is an independent set of $\Gamma_r^+(v)$. Further, all nodes of $\Gamma_r^+(v)$ and hence also all nodes of S are contained in a ball $B(v, r)$ of radius r around v. The number of balls of radius $1/2$ needed to cover $B(v, r)$ is at most $2^{\rho \log(2r)} = \mathrm{O}(r^\rho)$. □

One of the problems of the UDG model is that there is a sharp bound specifying whether two nodes hear each other. In [16, 83], a generalization of the UDG model relaxing this issue has been introduced. For general metric spaces, we obtain the following definition.

Definition 5.3. (Quasi Unit Ball Graph) *Let $M = (X, d)$ be a finite metric space and let $\lambda < 1$ be a constant. We defined a graph $G = (V, E)$ with node set $V = X$ and the following edge set E:*

- $d(u, v) \leq \lambda \implies (u, v) \in E$,
- $d(u, v) > 1 \implies (u, v) \notin E$.

If $\lambda < d(u, v) \leq 1$, it is not specified whether there is an edge between u and v. G is called a quasi unit ball graph.

Because every ball of radius $\lambda/2$ contains at most 1 node of an independent set of a quasi unit ball graph (QUBG) G, for constant λ and a metric M with constant doubling dimension, G is polynomially growth-bounded. If the underlying metric of a QUBG G is a sub-space of the 2-dimensional Euclidean plane, we call G a quasi unit disk graph (QUDG).

Metrics with bounded growth in general and doubling metrics in particular have found quite a lot of attention [59, 74, 76, 115, 126]. Besides UBGs and QUBGs, there are other network graphs which are based on such metric spaces. It is often assumed that latencies of many real networks such as peer-to-peer networks or the Internet are doubling. Although these network graphs are not growth-bounded in the sense of Definition 5.1, we will see that some of the results of this chapter also hold in this case. The network-related problems which have been considered for doubling metrics include metric embeddings [59, 74], distance labeling and compact routing [126], and nearest neighbor search [76]. The doubling dimension has been introduced in [59], however, a similar notion has already been used in [4].

5.2.2 Properties of Growth-Bounded Graphs

We conclude our introduction to growth-bounded graphs by looking at a few, in the context of this work important properties of such graphs. We have seen in Section 1.3.2 that every graph has a network decomposition with (strong) diameter $O(\log n)$ and chromatic number $O(\log n)$. Finding a good decomposition allows to devise efficient distributed algorithms for a great number of local problems. We will see in the following that every growth-bounded graph G has an $(O(1), O(1))$-decomposition, that is, G has a decomposition into clusters of constant diameter such that the cluster graph has constant chromatic number. In fact, we will see that we can even achieve cluster graphs with constant degree. We need the following definition from [112].

Definition 5.4. (r-Ruling Set) *Let $G = (V, E)$ be an undirected graph. A subset $R \subseteq V$ is called an r-ruling set of G if for each node $u \in V$, there is a node $v \in R$ with $d_G(u, v) \leq r$.*

If the nodes of an r-ruling set R are independent, that is, if no two nodes $u, v \in R$ are adjacent, we call R an r-ruling independent set. Note that 1-ruling independent sets are maximal independent sets. Assume that we are given an r-ruling independent set R of some f-growth-bounded graph G. The set R naturally induces a clustering of G. We make a cluster C_v for each node $v \in R$. Thereby, cluster C_v contains all nodes for which v is the closest node of R. Ties are broken arbitrarily. The following lemma shows that for constant r, the given clustering implies an $(O(1), O(1))$-decomposition.

Lemma 5.2. *The described clustering gives a $(f(2r+1)+1, 2r)$-decomposition of $G = (V, E)$. Moreover, the degree of the cluster graph is bounded by $f(2r+1)$. For constant r, we obtain an $(O(1), O(1))$-decomposition of G.*

Proof. The diameter of each cluster is at most $2r$ because R is an r-ruling set. It therefore suffices to show that the degree of the cluster graph is at most $f(2r+1)$. It then follows that the chromatic number of the cluster graph is at most $f(2r+1)+1$. Let us look at the neighbor clusters C_v of a cluster C_u for $u, v \in R$. Because for two neighboring clusters C_u and C_v there must be nodes $u' \in C_u$ and $v' \in C_v$ with $(u', v') \in E$, we have

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

Because R is an independent set, at most $f(2r+1)$ nodes $v \in R$ have distance at most $2r+1$ from u. □

In particular, Lemma 5.2 implies that every MIS S of a growth-bounded graph G induces an $(\mathrm{O}(1), \mathrm{O}(1))$-decomposition of G. Because the cluster graph given by S has constant degree, it can be colored in time $\mathrm{O}(\log^* n)$. Therefore, an MIS S can be converted into a $(\mathrm{O}(1), \mathrm{O}(1))$-decomposition in $\mathrm{O}(\log^* n)$ time on growth-bounded graphs. We have seen in Section 1.3.2 that from a (χ, d)-decomposition, an MIS can be computed in $\mathrm{O}(\chi d)$ time. Since rings are growth-bounded graphs, the $\Omega(\log^* n)$ lower bound for computing an MIS on a ring implies an $\Omega(\log^* n)$ lower bound on the number of rounds for computing an $(\mathrm{O}(1), \mathrm{O}(1))$-decomposition of a growth-bounded graph. Therefore, up to constant factors, constructing an MIS and computing an $(\mathrm{O}(1), \mathrm{O}(1))$-decomposition are equivalent problems on growth-bounded graphs. On the one hand, both structures cannot be constructed in fewer than $\Omega(\log^* n)$ rounds. On the other hand, one of the structures can be turned into the other one in $\mathrm{O}(\log^* n)$ time. In Section 5.3.3, we will see that this equivalence even holds in the $\mathcal{CONGEST}$ model if we additionally assume that the cluster graph of the decomposition has constant degree. In this case, we will show that based on an $(\mathrm{O}(1), \mathrm{O}(1))$-decomposition an MIS can be computed in $\mathrm{O}(1)$ time using messages of size $\mathrm{O}(\log n)$ only.

Lemma 5.2 shows that on growth-bounded graphs, an $\mathrm{O}(1)$-ruling set implies a decomposition with constant cluster diameter and degree. The following lemma shows that growth-bounded graphs are the only graphs with this property.

Lemma 5.3. *Let $\alpha \geq 1$ be a constant. Given an α-ruling independent set of a graph $G = (V, E)$, we construct a 2α-diameter decomposition of G as described above. If the degree of the obtained cluster graph is bounded by some constant $\beta \geq 2$ for all α-ruling independent sets of G, G is $2\beta^r$-growth-bounded.*

Proof. Consider some node $u \in V$. Assume that the decomposition is based on an α-ruling independent set R which contains a maximum independent set of the graph induced by the r-neighborhood $\Gamma_r^+(u)$ of u. Let C_u be the cluster containing u. The node $v \in R$ of a clusters C_v at distance at most r from C_u in the cluster graph is contained in $\Gamma_r^+(u)$. Because the degree of the cluster graph is bounded by β, it follows that $|\Gamma_r^+(u) \cap R| \leq \sum_{i=0}^{r} \beta^i \leq 2\beta^r$. □

5.3 Fast Deterministic MIS Construction

In Section 1.3, we have highlighted the importance of distributed MIS construction in the context of symmetry breaking. Section 5.2.2 shows that for the class of growth-bounded graphs, computing an MIS is even more fundamental because it can be used to directly obtain an $(O(1), O(1))$-decomposition of the network graph. Therefore, computing an MIS can be used as a basis to obtain efficient distributed algorithms for almost every local problem on growth-bounded graphs. In Section 1.3.2, we have seen that finding the distributed complexity for deterministically constructing an MIS is an important open problem. In particular, it is not known whether a poly-logarithmic time algorithm exists for general graphs. In this section, we answer this question in the affirmative for growth-bounded networks. We will present an algorithm which computes an MIS in time $O(\log \Delta \log^* n)$ in the $\mathcal{CONGEST}$ model. Hence, unless Δ is large, this even beats the $O(\log n)$-bound of the best randomized algorithms. The MIS algorithm for growth-bounded graphs consists of three parts which are described in Sections 5.3.1–5.3.3. In Section 5.3.1, we give an algorithm to compute an $O(\log \Delta)$-ruling independent set in time $O(\log \Delta \log^* n)$. Section 5.3.2 shows how, also in time $O(\log \Delta \log^* n)$, an $O(\log \Delta)$-ruling independent set can be turned into a 3-ruling independent set. Finally, Section 5.3.3 computes an MIS based on the network decomposition induced by the 3-ruling independent set of Section 5.3.2.

5.3.1 Constructing a Sparse Independent Set

The first phase of our MIS construction is a distributed algorithm which locally computes an $O(\log \Delta)$-ruling independent set S for a given undirected growth-bounded graph $G = (V, E)$ in time $O(\log \Delta \log^* n)$. A detailed description of the first phase is given by Algorithm 9. 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 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 9 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 $\overline{G}$, we can construct an MIS in time $O(\log^* n)$ [33, 55, 91]. 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

Algorithm 9 Computing a sparse independent set (code for vertex v)

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

which $d(p(u)) = u$. The edge set of $\overline{V}$ 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 5.1).

Lemma 5.4. *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 methods described in [33, 55, 91].

Lemma 5.5. *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 until the next while-loop iteration. Node v can

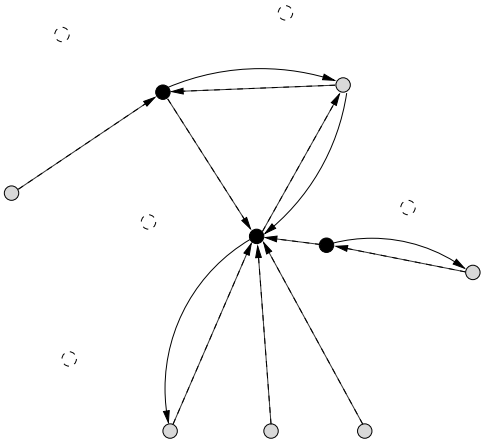

Figure 5.1: One iteration of Algorithm 9. The dashed nodes are passive at the outset of the iteration. The dashed arrows between active nodes denote the links $d(v)$. The graph $\overline{G}$ is induced by the links $p(v)$ which are denoted by the solid, bended arrows. Finally, the algorithm computes an MIS on $\overline{G}$, leaving only the black nodes active for the next iteration.

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 of 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 can choose a node $u = d(v)$ in Line 5. Since $A_u \neq \emptyset$, u chooses a node $p(u)$ and therefore 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 of u is still active after completing the while-loop iteration. Since $\mathrm{dist}_G(v, w) = 2$, this completes the proof. □

The following two lemmas give bounds on the number of rounds needed by Algorithm 9 to complete. Further, the resulting structure is described. Lemma 5.6 bounds Algorithm 9 for general graphs, whereas Lemma 5.7 gives bounds for the algorithm when applied to growth-bounded graphs.

Lemma 5.6. *On any graph G, Algorithm 9 terminates with an $\mathrm{O}(\log n)$-ruling independent set S after $\mathrm{O}(\log n)$ consecutive 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 $n_{\mathrm{act}}/3$ nodes become passive. The claim then follows by Lemma 5.5.

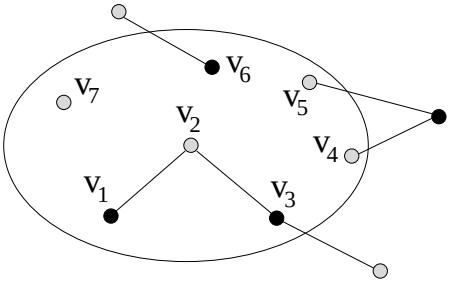

Figure 5.2: The cluster with the edges in $\overline{G}$. Black nodes will remain active in the next iteration. The nodes v_1, v_2, and v_3 are in C_i. Nodes v_4, v_5, and v_6 are connected only to nodes outside of the cluster and hence, are in set C_o. Finally, $v_6 \in C_p$.

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. However, $\overline{G}$ is constructed such that it does not contain isolated nodes, that is, all nodes of $\overline{G}$ have at least degree 1. Note that $\overline{G}$ is an edge-induced subgraph of G. Because the maximum degree of a node in $\overline{G}$ is 2 (Lemma 5.4), the MIS I consists of at most $2\overline{n}/3$ nodes. Hence, at least $\overline{n}/3$ nodes become passive in Line 17. □

For general graphs, a slightly faster algorithm to compute an $O(\log n)$-ruling independent set has been presented in [9]. The algorithm of [9] needs $O(\log n)$ rounds and is therefore by a factor of $O(\log^* n)$ faster. However, as the following lemma shows, Algorithm 9 is faster than the algorithm of [9] for growth-bounded graphs.

Lemma 5.7. *If the network graph G is growth-bounded, after $O(\log \Delta)$ consecutive executions of the while-loop, Algorithm 9 terminates with a $O(\log \Delta)$-ruling independent set S.*

Proof. Let M be a maximal independent set of G. The set M defines a clustering as described in Section 5.2.2. 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. Let us define the cluster graph G_C as follows. The nodes of G_C are the clusters C_u. Two nodes C_u and C_v are connected if there is an edge connecting the respective clusters. Because we assume that G is growth-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 G_C is bounded by $d := f(3)$.

In the following, we show that the maximum number of active nodes per cluster is reduced by a factor of 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. Let α be the maximum number of active nodes per cluster at some time t. We will 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. Let us look at a single iteration of the while-loop of Algorithm 9. We partition the c active nodes of C_u into three groups according to their neighbors in $\overline{G}$ (Figure 5.2). 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 $\mathcal{C}_p$. The set of nodes which have a neighbor inside C_u and which are only connected to nodes outside C_u are called $\mathcal{C}_i$ and $\mathcal{C}_o$, respectively. Clearly, we have $|\mathcal{C}_p| + |\mathcal{C}_i| + |\mathcal{C}_o| = c$. Because the maximum degree of $\overline{G}$ is 2, during the construction of the MIS in Line 10 at least one third of the nodes in $\mathcal{C}_i$ become passive. The nodes in $\mathcal{C}_o$ can be divided into the nodes $\mathcal{C}_o^p$ which become passive and the nodes $\mathcal{C}_o^a$ which stay active. Each node outside C_u is connected to at most 2 nodes in $\mathcal{C}_o^a$. Therefore, at least $|\mathcal{C}_o^a|/2$ nodes outside C_u become passive. Let $c_i := |\mathcal{C}_p| + |\mathcal{C}_i| + |\mathcal{C}_o^p|$ and $c_o := |\mathcal{C}_o^a|$. 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 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

$$\frac{1}{3} \cdot \sum_{j=1}^{k} c_i^{(j)} \leq \frac{\alpha}{2} \tag{5.1}$$

because otherwise at least $\alpha/2$ nodes of C_u would have become passive. Therefore, the number of nodes in the neighbor clusters of C_u which have become passive is at least

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

Because of Equation (5.1), this is at least $(k-3)\alpha/4$. As there are at most $d\alpha$ active nodes in neighbor clusters of C_u at the beginning, after $\mathrm{O}(d) = \mathrm{O}(1)$ iterations of the while-loop there are no active nodes left in the neighborhood of C_u. From then on, at least one third of the nodes in C_u becomes passive in every further iteration. □

Algorithm 10 Computing a dense independent set

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: **od**;
8: **od**

Summarizing Lemmas 5.4–5.7, we obtain the following theorem.

Theorem 5.8. *Algorithm 9 is a local, distributed algorithm which computes an* $\mathrm{O}(\log \Delta)$*-ruling independent set in* $\mathrm{O}(\log \Delta \cdot \log^* n)$ *rounds for any growth-bounded graph* $G = (V, E)$*. For general graphs, the algorithm terminates in* $\mathrm{O}(\log n \cdot \log^* n)$ *rounds producing an* $\mathrm{O}(\log n)$*-ruling independent set. All messages are of size* $\mathrm{O}(\log n)$*.*

5.3.2 Making the Independent Set Dense

In the following, we show how the relatively sparse independent set which we constructed so far can be made dense enough to obtain an $(\mathrm{O}(1), \mathrm{O}(1))$-decomposition for growth-bounded graphs. Specifically, we show how on growth-bounded graphs, a t-ruling independent set can be transformed into a 3-ruling independent set in $\mathrm{O}(t \log^* n)$ rounds using messages of size $\mathrm{O}(\log n)$. Algorithm 10 describes the basic method to achieve this. The idea is to enlarge the independent set in small steps such that it gets denser in each step. Before coming to a detailed analysis, we give a rough overview. In Line 3, 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 4 and 5, the independence of the extended independent set is restored by computing an MIS on the new nodes (see Lemma 5.10). 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 5.9. *Let* S' *be a* t*-ruling independent set for* $t > 3$*. After one iteration of the while-loop of Algorithm 10,* 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. However, this is clearly guaranteed because in Line 6 a maximal independent set 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, we 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 of y join the independent set S'. The distance between v and y is $t-1$ and the distance between v and z is $t-1$, which concludes the proof. □

It remains to be shown that Algorithm 10 can indeed be implemented by an efficient distributed algorithm. Lemma 5.10 gives exact bounds on the distributed complexity of Algorithm 10.

Lemma 5.10. *Let G be a growth-bounded graph. On G, Algorithm 10 can be executed by a distributed algorithm with time complexity $\mathrm{O}(t \log^* n)$ using messages of size $\mathrm{O}(\log n)$.*

Proof. By Lemma 5.9, Algorithm 10 terminates after at most t iterations of the while-loop. We therefore have to prove that each while-loop iteration can be executed in time $\mathrm{O}(\log^* n)$ using messages of size $\mathrm{O}(\log n)$. 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 a 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 a 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 highest ID joins $\hat{S}_u$. Because we assume that G is a growth-bounded graph, there can be at most $f(4) = \mathrm{O}(1)$ independent nodes in $\Gamma_u^+(u)$ 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 be proved that the construction of the MIS in Line 6 of Algorithm 10 can be computed in $\mathrm{O}(\log^* n)$ rounds. Let us therefore have a look at the structure of the graph $\mathcal{G}$ which is induced by the union of the sets $\hat{S}_u$ for all $u \in S'$. Consider a node v of $\mathcal{G}$, that is, $v \in \hat{S}_u$ for some $u \in S'$. Further, let w be a neighbor of v in $\mathcal{G}$. The 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 a growth-bounded graph, there exists a function f such that there are at most $f(5)$ independent nodes at distance at most 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 $f(5) \cdot f(1) = \mathrm{O}(1)$. We know that on a constant-degree graph, an MIS can be constructed in $\mathrm{O}(\log^* n)$ rounds using messages of size $\mathrm{O}(\log n)$. □

Combining Lemmas 5.9 and 5.10 we obtain the next theorem.

Theorem 5.11. *On a growth-bounded graph, a t-ruling independent set can be transformed into a 3-ruling independent set in $\mathrm{O}(t \log^* n)$ rounds using messages of size $\mathrm{O}(\log n)$.*

5.3.3 Computing the MIS

We will now describe the last phase of our algorithm, turning the 3-ruling independent set S' from Algorithm 10 into an MIS. As described in Section 5.2.2, S' induces a natural clustering of the nodes of G. Because S' is a 3-ruling set, the distance between the centers u and v of two neighboring clusters C_u and C_v can be at most 7. The degree of the cluster graph $\mathcal{G}_{S'}$ is therefore bounded by $f(7) = \mathrm{O}(1)$ if G is f-growth-bounded. The first step of the third phase of our MIS algorithm is to compute $\mathcal{G}_{S'}$ and to color $\mathcal{G}_{S'}$ with $f(7)+1$ colors, resulting in a $(\mathrm{O}(1), \mathrm{O}(1))$-decomposition of G. Applying algorithms from [33, 55, 91], this can be achieved in $\mathrm{O}(\log^* n)$ rounds using messages of size $\mathrm{O}(\log n)$.

Having computed this decomposition, we can now compute an MIS M of G by sequentially computing the contributions from each color of the coloring of $\mathcal{G}_{S'}$. For each node v, let x_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 $x_u < x_v$ or if $x_u = x_v \wedge \mathrm{ID}(u) < \mathrm{ID}(v)$. Each node now proceeds as follows. Initially, we set $M = S'$. All nodes v of S' inform their neighbors about the joining of 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 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. As shown by the next lemma, the described algorithm computes an MIS M in time $\mathrm{O}(1)$.

Lemma 5.12. *On f-growth-bounded graphs the above algorithm computes an MIS M in time $2f(7)f(3)$.*

Proof. We first show that M indeed is an independent set of G. For the sake of contradiction, assume that there are two adjacent nodes u and v which both join M. W.l.o.g., we assume that $u \prec v$. Assuming that v joins M means that v must have received a COVERED(u) message from u. However, this is a contradiction to the assumption that u joins M. To see that M is an MIS, observe that as long as M is not maximal, there is a smallest node u (with respect to $\prec$) which is not covered.

What remains to be proved the time complexity of the above algorithm. First note that because the radius of each cluster is at most 3, there can be at most $f(3)$ MIS nodes per cluster. Let us now look at a single cluster C_u of the smallest color 1. Because with respect to the order $\prec$ the nodes of $\mathcal{C}_u$ 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 there are at most $f(3)$ nodes of C_u which 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. □

Combining Theorems 5.8 and 5.11 as well as Lemma 5.12, we obtain the following upper bound on the complexity of the deterministic MIS construction on growth-bounded graphs.

Theorem 5.13. *Let G be a growth-bounded network graph. There is a deterministic distributed algorithm which constructs a maximal independent set on G in time $\mathrm{O}(\log \Delta \cdot \log^* n)$ in the $\mathcal{CONGEST}$ model.*

5.4 Algorithms Based on Coordinates or Distances

As discussed in Section 5.1.1, wireless ad hoc and sensor networks are often modeled as unit disk graphs. Additionally, many algorithms assume that nodes have access to coordinate information [1, 3, 82, 129] or that nodes can measure distances or angles to neighboring nodes [51, 131, 132]. Coordinates are obtained from a positioning system such as GPS either directly or by running a distributed positioning algorithm [21, 25, 121]. In this section, we will have a closer look at this graph model in the context of local algorithms. In Sections 5.4.1 and 5.4.2, we have a look at the UDG model with coordinate and internode distance information, respectively. In Section 5.4.1, we described how an $(\mathrm{O}(1), \mathrm{O}(1))$-decomposition is essentially obtained for free if nodes know their coordinates. We show in Section 5.4.2 that if nodes have no access to their coordinates but can find out the distances to their neighbors, the main ideas

1	2	3	4	1	2	3	4
7	8	5	6	7	8	5	6
3	4	1	2	3	4	1	2
5	6	7	8	5	6	7	8
1	2	3	4	1	2	3	4

Figure 5.3: Coloring of the grid with 8 colors

of Section 5.4.1 can still be used to compute constant approximations for covering and packing problems in a constant number of rounds. In Section 5.4.3, we consider the more general unit ball graph model. If adjacent nodes know their distance with respect to the underlying metric, it is possible to compute an $(\mathrm{O}(1), \mathrm{O}(1))$-decomposition in $\mathrm{O}(\log^* n)$ rounds on a doubling UBG. We will see that the results of Section 5.4.3 also hold for other network models based on doubling metric spaces or if distances are only known up to a constant factor.

5.4.1 Global Coordinates

Let us now consider the case where the network graph G is a UDG and where all nodes of G know their coordinates. Note that this means that all nodes 'see' a common global coordinate system. Having a global coordinate system enables to compute a decomposition as follows. We partition the plane by a grid into square cells of side length $1/\sqrt{2}$. Each square cell defines a cluster of nodes. By checking their coordinates, nodes can decide in which cell they are located and hence to which cluster they belong. Since the length of the diagonal of a single square cell is 1, the induced graph of 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 Euclidean distance is at most 1 are colored with the same color. Figure 5.3 shows how this can be achieved using 8 colors. Hence, by assigning each cluster the respective color, we obtain a $(8, 1)$-decomposition.

It is of course not surprising that global information such as coordinates helps devising fast distributed algorithms. The possibility of computing a $(\mathrm{O}(1), \mathrm{O}(1))$-decomposition from UDG coordinates alone indicates the power of such coordinate information. In other workds, unit disk graph coordinates suffice to compute essentially everything which can be computed locally in a constant number of rounds. In the next section, we will see that the described simple algorithm for computing a network decomposition can even be applied in some form in the absence of global information.

5.4.2 Fractional Covering and Packing Problems

In most cases, it is not realistic to assume that there is a positioning system which nodes can use to obtain coordinate information. In this section, we show that the main ideas of the last section can be adapted in order to solve many interesting problems in a case where no global information is present.

We again consider the standard unit disk graph model. In addition to knowing the direct neighbors, we assume that nodes can sense the distances to their neighbors. By exchanging this information for a few rounds, this enables the nodes to build up a *local coordinate system*. That is, distances between nodes can be used to compute angles and to learn about the geometry of the neighborhood. It is however not possible to align all these local coordinate systems; each node has its own local view. Assume for instance that we want to compute a small dominating set. In the presence of global coordinates, we can compute a network decomposition as described in the last section. Choosing one node per cluster (e.g., the node with the largest ID) gives a dominating set which is only by a constant factor larger than an optimal dominating set. If we try to do this with the local coordinate systems, the clusters of different local systems will be different. Hence, also the selected nodes (dominating set) will be different in each coordinate system. This can lead to disastrous solutions and does not yield a non-trivial approximation.

While all the local coordinate systems inherently differ from each other, the set of all possible global coordinate systems is the same at every node. Hence, if we computed all dominating sets corresponding to the clusterings of all (infinitely many) different global coordinate systems, all nodes would come up with their local part of the same (multi-)set of different global dominating sets. It is of course still not possible to globally select one of these dominating sets. However, if we assign values 0 and 1 to non-dominators and dominators, respectively, it is possible to compute the *average* over all dominating sets. This does not result in a global dominating set, however it does result in a fractional dominating set solution, that is, we solve the natural LP relaxation ($\mathrm{LP}_{\mathrm{DS}}$) of the dominating set problem. In the following, we present an explicit and more general algorithm for the above intuitive description. The algorithm can be applied to general covering LPs (PP) and packing LPs (DP) as introduced in Section 2.1.3.

Because we restricted the communication graph G to be a UDG, we cannot use the bipartite network graph of Section 2.1.3. Instead, we assume that all primal variables x_i and all dual variables y_i represent some value in the graph, that is, they belong to some node or edge of G. We assume that the conditions of the LP are local in the sense that whenever a primal (dual) variable x_i (y_j) occurs in the inequality corresponding to a dual (primal) variable y_j (x_i), x_i and y_j are separated by at most a constant number of hops in the network graph. This locality condition is true in all natural network coordination problems such as minimum dominating set (MDS), maximum matching (MM), etc. Remember that in MDS, for each node v_i, there is a primal variable x_i and a dual variable

y_i. The primal feasibility condition demands that the sum of the x-values in the 1-neighborhood of all nodes is at least 1, the dual feasibility is achieved if the sum of the y-values of each 1-neighborhood is a most 1.

We will now first look at a solution of such LPs based on the network decomposition of Section 5.4.1. We will then show how to convert this into a solution which does not need global coordinates using the idea of averaging over the set of all possible solutions.

Assume that we are given a $(O(1), 1)$-decomposition as described in Section 5.4.1. By exchanging the IDs among direct neighbors, each cluster can select the node with the largest ID as leader. In parallel, each leader then computes a local LP such that the combined local solutions form a constant approximation for (PP) or (DP). The local LPs are computed as in Section 2.4. Let v_0 be the leader of some cluster C_{v_0}. Let $\mathcal{Y}_0$ be the set of all dual y-variables which belong to nodes at distance at most 1 from v_0 or to edges which are adjacent to neighbors of v_0. The set $\mathcal{Y}_0$ has a corresponding set $\mathcal{E}_0$ of primal inequalities of (PP). Let $\mathcal{X}_0$ be the set of primal x-variables which occur in the inequalities $\mathcal{E}_0$. We define P_0 to be the covering problems consisting of the objective function of (PP) and the inequalities $\mathcal{E}_0$. P_0 is an LP on the variables $\mathcal{X}_0$. The packing problem D_0 is obtained by deleting all variables from (DP) which are not in $\mathcal{Y}_0$. That is, we restrict the matrix A to the rows and columns defined by $\mathcal{Y}_0$ and $\mathcal{X}_0$, respectively. By the definition of (PP) and (DP), the nodes and edges of the variables in $\mathcal{X}$ and $\mathcal{Y}$ are all within constant distance from v_0. Thus, v_0 can locally solve P_0 and D_0 in a constant number of rounds. The local solutions for each cluster can be combined by summing up the values of all local LPs for each variable.

Lemma 5.14. *Summing up the described local LPs for all clusters yields solutions for* (PP) *and* (DP) *with the same value of the objective function. The solution of* (PP) *is a feasible constant approximation, the solution of* (DP) *can be made feasible by dividing each y-variable by a constant factor.*

Proof. We start by proving the feasibility of (PP). Because all clusters have diameter 1, all dual y-variables are in the set $\mathcal{Y}_i$ of at least one cluster leader v_i. Therefore, every inequality of (PP) occurs in some local LP P_i. Because the x-values of all local LPs are summed up, it is sufficient to make every primal covering constraint feasible once in order to obtain a globally feasible solution for (PP).

For the almost-feasibility of (DP), observe that we have chosen the set $\mathcal{Y}_0$ such that the solution of the local dual problem D_0 is feasible for (DP) (set all unused y-variables to 0): Clearly all inequalities which appear in D_0 are also feasible for (DP); because all inequalities of (DP) containing a variable $y_i \in \mathcal{Y}_0$ also appear in D_0, all other inequalities of (DP) are of the form $0 \leq c_j$ for some j. Because of the locality condition for our LPs, all $x_i \in \mathcal{X}_0$ are at a constant distance from v_0. Therefore, each x_i can only occur in a constant number of local covering LPs. Because there is a one-to-one correspondence between primal variables and dual inequalities, each dual inequality can as well

only occur in a constant number of local LPs. Because each local LP is dual-feasible for (DP), this means that the sum of all local LPs is dual feasible for (DP) up to a constant factor.

If all local LPs are solved optimally, the values of the objective functions for a pair (P_0, D_0) of local LPs are equal. Therefore, when summing up the local LPs, we get the same objective function values for (PP) and (DP) as well. By LP duality, the approximation factor of (PP) is at most equal to the constant factor by which the dual inequalities have to be divided in order to obtain a feasible (DP) solution. □

We will now show how to average the described solution over all possible coordinate systems. Equivalently to averaging the x and y values for all possible coordinate systems, we can choose one coordinate system uniformly at random[1] and compute the *expected values* for the x and y variables. In the above description, we have chosen the local LPs such that they are independent of the nodes' assignments to clusters. They only depend on the choice of the cluster leaders. Hence, each node v_i can compute its local LP. Let p_i be the probability that v_i is a cluster leader if the coordinate system is chosen uniformly at random. If we assume that every node v_i can compute its p_i, a constant-factor approximation to a given covering or packing LP can be computed as follows.

1. compute local LP and p_i
2. increase all variables x_j or y_j of LP by $p_i x_j$ or $p_i y_j$, respectively
3. if LP is a packing problem, divide by appropriate constant factor

It remains to be shows that p_i can really be computed. We will present an elegant way to approximate p_i up to a small constant factor. By the construction of the network decomposition of Section 5.4.1, p_i is the probability that v_i is the node with the largest ID within its cell of a random square grid of cell size $1/\sqrt{2}$. Hence, p_i is the probability that v_i has the largest ID in a random square of side length $1/\sqrt{2}$ containing v_i. Because for every square of side length $1/\sqrt{2}$, there is a circle of diameter $1/\sqrt{2}$ which is completely inside the square, the probability p_i' of having the largest ID in a random circle of diameter $1/\sqrt{2}$ is $p_i' \geq p_i$. Using p_i' instead of p_i in the above algorithm therefore guarantees that the computed (PP) solution is feasible.

We will now argue that the objective functions are not affected too much by using p_i' instead of p_i. Let p_i'' be the probability that v_i is the node with largest ID in a square of side length $1/2$ containing v_i. Taking p_i'' instead of p_i corresponds to making the network decomposition with a grid of cell size $1/2$ instead of $1/\sqrt{2}$. Using p_i'' in the above algorithm would therefore give a solution which is at most by a factor of 2 worse than the solution when using p_i because the area of each cell is smaller by a factor of 2. Hence, the number of

[1] In principle, this means that the origin and the direction of the x-axis are chosen uniformly at random

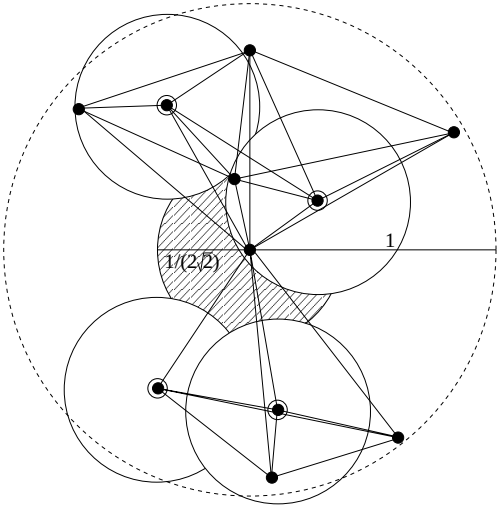

Figure 5.4: Computation of p'_i: Node v_i is at the center, the encircled nodes are the neighbors of v_i having a larger ID than v_i. The shaded area is proportional to p'_i.

neighboring clusters in the decomposition doubles. Additionally, we have that $p''_i \geq p'_i$ because every circle of diameter $1/\sqrt{2}$ completely contains a square with side length $1/2$. Thus, taking p'_i instead of p_i in the described algorithm results in a feasible solution for (PP) which is worse than the solution using p_i by at most a factor of 2.

The probability p'_i can be computed by v_i as follows.

1. exchange 1-hop distances with neighbors
2. compute angles between adjacent neighbors
3. geometrically arrange neighbors in one possible way.
4. For some node v, let $D(v)$ be the disk with radius $1/(2\sqrt{2})$ around v. Further, let $N^+(v)$ be the set of neighbors of v which have a larger ID than v. The probability p'_i can be computed as the area of

$$D(v_i) \setminus \bigcup_{u \in N^+(v_i)} D(u)$$

divided by the area of $D(v_i)$. That is, p'_i is the fraction of $D(v_i)$ which is not covered by any of the disks $D(u)$ with $\mathrm{ID}(u) > \mathrm{ID}(v_i)$.

Figure 5.4 illustrates step 4 of the described algorithm.

Lemma 5.15. *The above algorithm correctly computes the probability p'_i that v_i is the node with the largest ID in a random circle of diameter $1/\sqrt{2}$ containing v_i.*

Proof. We first assume that step 3 of the algorithm is unique, that is, we have a local coordinate system where v_i and its neighbors are correctly geometrically arranged. Choosing a random circle of diameter $1/\sqrt{2}$ containing v_i can be done by placing the center of the circle uniformly at random in the disk of radius $1/(2\sqrt{2})$ around v_i. For a given center p, v_i has the largest ID if there is no node v_j with $\mathrm{ID}(v_j) > \mathrm{ID}(v_i)$ at distance at most $1/(2\sqrt{2})$ from p. Therefore, v_i does have the largest ID if and only if the center p is chosen at distance more than one from all neighbors v_j of v_i with $\mathrm{ID}(v_j) > \mathrm{ID}(v_i)$. This is exactly the case if p is in the area which is computed in step 4 of the algorithm. Hence, the lemma is true if step 3 is unique.

Let $N(v_i)$ be the induced graph of v_i's neighbors (not including v_i), that is, the edges of $N(v_i)$ are all edges between neighbors of v_i. We start with the case where $N(v_i)$ consists of a single component. If we know the distances to two adjacent neighbors as well as the distance between those two neighbors, we can compute the angle at v_i between the two neighbors. If $N(v_i)$ is a single component, we can then find the angles between all neighbors of v_i. The geometry of the 1-neighborhood of v_i is therefore determined up to rotation around v_i, that is, we can compute a local coordinate system for which step 3 of the algorithm is unique.

Step 3 of the algorithm is not unique if $N(v_i)$ consists of several connected components. The geometry of each component can be determined, however the angle between different components cannot be inferred from the knowledge of the distances between neighbors alone. However, two nodes from different components of $N(v_i)$ are at distance more than 1 from each other. Therefore, the disks of radius $1/(2\sqrt{2})$ around two nodes $u, u' \in N(v_i)$ do not intersect if u and u' belong to different components in $N(v_i)$. Thus, the area which is computed in step 4 is the same for all possible geometric arrangements of the neighbors of v_i. □

The results of this section are summarized in the upcoming theorem. The time bound for the fractional dominating set problem follows because in this case the given algorithm becomes particularly simple. The local LPs can be solved by assigning 1 to all cluster leaders and 0 to all other nodes. Thus, the values p'_i form a constant approximation for minimum fractional dominating set.

Theorem 5.16. *In the given UDG model where distances are known, all local fractional covering and packing problems can be approximated up to a constant factor in constant time. In particular, the fractional minimum dominating set problem can be approximated in a single round.*

Combined with the randomized rounding algorithms presented in Section 2.5, Theorem 5.16 allows to compute approximate solutions for many integer

covering and packing problems in constant time. In particular, in a single round, it is possible to compute a dominating set with expected approximation ratio $O(\log \Delta)$ if the network graph is a UDG and if all nodes know the distances to their neighbors.

To conclude this section, we would like to highlight an intriguing comparison concerning the distributed complexity on unit disk graphs and on general graphs. In Chapter 3, we have shown that on general graphs, approximating fractional covering and packing problems up to a constant factor needs time at least $\Omega(\sqrt{\log n / \log \log n})$. The fact that in the given unit disk graph model we can compute a constant approximation in a single communication round shows that there can be a large gap between the distributed complexity of problems on the unit disk graph and on general graphs.

5.4.3 Network Decomposition

We have seen that for the unit disk graph knowing the distances to direct neighbors is sufficient to reasonably approximate important problems such as minimum dominating set in just one round or a constant number of rounds. If we want to compute more sophisticated structures such as a maximal independent set or an $(O(1), O(1))$-decomposition, the methods of Section 2.3 cannot be used. Based on the $\Omega(\log^* n)$-lower bound for the ring [91], it is in fact not hard to see that it is not even possible to construct an MIS or a decomposition in a constant number or rounds. Because a ring where all edges have length 1 is a unit disk graph, this lower bound applies to our model. Note that it does not help to know the edge lengths if all edges have length 1. In this section, we will show that in the model of Section 2.3, it is indeed possible to compute an $(O(1), O(1))$-decomposition in $O(\log^* n)$ rounds. Our result even holds if the network graph is a doubling unit ball graph.

Basic Algorithm

We will first present a (potentially slow) deterministic distributed algorithm which computes a $(O(1), 2)$-decomposition. In a second step (Section 5.4.3), we will then show how the algorithm can be implemented such that its runtime is $O(\log^* n)$ in the $\mathcal{LOCAL}$ model. For the slow version of the algorithm, we assume that all nodes know the minimum distance $d_{\min}$ between any two nodes. This assumption would not be necessary. However, making the assumption results in a simpler, easier to understand algorithm. For the fast implementation, the assumption is not needed anymore. The computing of the decomposition is described by Algorithm 11.

Algorithm 11 starts with a small radius r which is increased by a factor of 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, a subset of the nodes is selected such that the nodes selected in the subset form a maximal independent set on the graph of all edges of length $\leq r$.

Algorithm 11 Network Decomposition: Clustering

1: $r := \min\{2^\lambda | \lambda \in \mathbb{Z} \wedge 2^\lambda \geq d_{\min}\}$;
2: $\mathcal{V} := V$;
3: **while** $r \leq 1/2$ **do**
4: $\quad \mathcal{G} := (\mathcal{V}, \mathcal{E})$ with $\mathcal{E} = \{\{u, v\} | d(u, v) < r\}$;
5: $\quad$ compute MIS on $\mathcal{G}$;
6: $\quad \mathcal{V} := \{v \in \mathcal{V} | v \text{ in MIS}\}$;
7: $\quad r := r \cdot 2$
8: **od**;
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 G_C. Color G_C with $\Delta_C + 1$ colors.

Lemma 5.17. *Algorithm 11 computes a $(2^{4\rho}, 2)$-decomposition where ρ is the doubling dimension of the underlying metric. The maximum degree of the cluster graph is at most $2^{4\rho} - 1$.*

Proof. We first prove that each node has a cluster leader at distance at most 1 and that therefore the diameter of each cluster is at most 2. The algorithm maintains a set $\mathcal{V}$ of nodes which are cluster leader candidates. In each iteration, some nodes are removed from $\mathcal{V}$. We have to prove 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 a cluster leader. Let $r_u = 2^{\lambda_u}$ ($\lambda_u \in \mathbb{Z}$) be the radius at which u is removed from $\mathcal{V}$. Whenever a node is removed from $\mathcal{V}$, there is a node at distance at most r which stays in $\mathcal{V}$. Otherwise, the independent set which is computed in Line 5 is not maximal. 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 $\mathcal{V}$. We thus 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, we therefore get

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

where r_{u_i} is the radius of the iteration where 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 11 is $1/2$, we have $d(u, v) < 1$. Thus, the radius of each cluster is at most 1.

It now remains to be shown that the maximum degree Δ_C of the cluster graph is at most $2^{4\rho} - 1$. On the one hand, from the last iteration of the algorithm ($r = 1/2$), it is guaranteed that the distance between any two cluster leaders is more than $1/2$. Otherwise, the nodes of the MIS of Line 5 would not

be independent. Therefore, 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 in $B_3(v)$, the ball with radius 3 around v. By the definition of ρ, $B_3(v)$ can be covered by at most $2^{4\rho}$ balls of radius $3/16 < 1/4$. Including v, the number of cluster leaders in $B_3(v)$ is therefore at most $2^{4\rho}$. □

We will now have a close look at the complexity of a single iteration of the while-loop of Algorithm 11. From a complexity point of view, the most important part is the computation of the MIS in Line 5. Everything else (computing the neighbors in $\mathcal{G}$ and informing neighbors about new $\mathcal{V}$) can be done in a constant number of rounds. On networks of maximum degree Δ, an MIS can be computed in time $O(\Delta \log \Delta + \log^* n)$ (see Chapter 4); that is, for constant-degree networks constructing an MIS needs $O(\log^* n)$ rounds. The following lemma shows that the maximal node degree of $\mathcal{G}$ is indeed bounded.

Lemma 5.18. *In each iteration of Algorithm 11, the maximum degree of $\mathcal{G}$ is at most $2^{2\rho}$.*

Proof. Let ℓ denote the minimum distance between any two nodes of $\mathcal{G}$. 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. All neighbors of a node $v \in \mathcal{V}$ are in the ball $B_r(v)$ of radius r around v. By the definition of the doubling dimension ρ, $B_r(v)$ can be covered by $2^{2\rho}$ balls of radius $r/4$. Therefore, the number of nodes in $B_r(v)$ is at most $2^{2\rho}$. □

Lemma 5.18 implies the following corollary.

Corollary 5.19. *The time complexity of a single iteration of the while-loop of Algorithm 11 is $O(\log^* n + \rho 2^{2\rho})$, that is, for constant doubling dimension the time complexity is $O(\log^* n)$.*

Before coming to the description of a faster implementation of the while-loop of Algorithm 11, we have a look at the complexity of Lines 9 and 10. By Lemma 5.17, we know that each node has a cluster leader in its neighborhood. Line 9 thus can be computed in a single communication round. The time complexity of Line 10 is more interesting. We have seen in Chapter 4 (Theorem 4.8) that a $(\Delta+1)$-coloring can be constructed in $O(\Delta \log \Delta + \log^* n)$ rounds. We therefore get the following lemma.

Lemma 5.20. *The time complexity of Line 10 of Algorithm 11 is $O(\log^* n + \rho 2^{4\rho})$, that is, for constant doubling dimension the time complexity is $O(\log^* n)$.*

Fast Implementation of the Basic Algorithm

In this section, we will have a second look at Algorithm 11 leading to a better time complexity. We need to start with a few general considerations concerning the synchronous message passing model. If nodes communicate for k rounds, they can only gather information which is at most k hops away. In principle, every distributed k-round algorithm can be formulated as follows.

1. Collect complete k-neighborhood of graph in k communication rounds.

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

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. The above formulation of a distributed algorithm of course has the drawback that messages can get extremely large. We will show that the message size can be kept moderate in our example.

Let us again consider a single iteration of the while-loop of Algorithm 11. All communication which is needed to compute an iteration of the while-loop is on $\mathcal{G}$. Hence, all messages are sent on edges which have length at most r. If we communicate for k rounds and if all messages of those k rounds are on edges of length at most r, then all collected information comes from distance at most $k \cdot r$. In order to be able to compute everything locally, the nodes have to collect the complete neighborhood up to distance kr (w.r.t. the metric). That is, the nodes have to collect all information which is accessible by paths of length at most kr. Note that it is not necessary and it might not be possible to collect the whole ball of radius kr. Because of the triangle inequality, it is possible to collect this information in $2kr$ rounds. Applying this to Algorithm 11, we get Lemma 5.21.

Lemma 5.21. *Algorithm 11 can be computed in* $\mathrm{O}(\log^* n + \rho 2^{4\rho})$ *rounds, that is, for constant doubling dimension the time complexity can be reduced to* $\mathrm{O}(\log^* n)$.

Proof. By Corollary 5.19, the number of rounds of an iteration of the while-loop of Algorithm 11 is $\mathrm{O}(\log^* n + \rho 2^{2\rho})$. Nodes therefore need to collect information from distance at most $\mathrm{O}(r(\log^* n + \rho 2^{2\rho}))$. To obtain the distance from which we need information in order to be able to locally compute the results of all iterations of the while-loop, we have to sum up the distances for all iterations. We do not know the number of 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 twice the maximum summand. Therefore, the whole while-loop can be computed in $\mathrm{O}(\log^* n + \rho 2^{2\rho})$ rounds. Together with Lemma 5.20, we get the required result. □

Note that when collecting the whole neighborhood, it is not necessary that nodes know the minimum distance $d_{\min}$ between nodes. Because the radius grows exponentially, the locality of the problem is independent of the starting radius. Each node can just use the smallest distance in the collected neighborhood in order to locally simulate the distributed Algorithm 11. The complete algorithm to compute a $(\mathrm{O}(1), \mathrm{O}(1))$-decomposition in the given network model can be summarized as follows.

1. Exchange 1-hop distances with neighbors.
2. Locally compute the while-loop of Algorithm 11 for $r \in \mathrm{O}(1/(\log^* n + \rho 2^{2\rho}))$ (up to the radius for which it suffices to know the 1-neighborhood).
3. Collect $\mathrm{O}(\log^* n + \rho 2^{2\rho})$-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 and 10 of Algorithm 11).

Computing the solution for small radii first and then collecting the rest of the neighborhood is done in order to obtain reasonable message sizes. We are now ready to formulate our main theorem.

Theorem 5.22. *In the unit ball graph model, the above algorithm computes a* $(2^{4\rho}, 2)$*-decomposition in time* $\mathrm{O}(\log^* n + \rho 2^{4\rho})$ *where* ρ *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*

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

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

Proof. The time complexity follows from Lemma 5.21. For the correctness of the algorithm it remains to be proved that only collecting information about nodes in $\mathcal{V}$ for $r \geq \mathrm{O}(1/(\log^* n + \rho 2^{2\rho}))$ (Steps 3 and 4) is sufficient. Because all communication of Algorithm 11 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 $\mathrm{O}(\Delta \cdot K)$. In Step 3, a message can contain at most the whole R-neighborhood of a node, where $R := \mathrm{O}(\log^* n + \rho 2^{2\rho})$. Let N be the maximum number of nodes which such an R-neighborhood can contain. If $r \in \Theta(1/(\log^* n + 2^{4\rho}))$ 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,

balls of radius at most $r/2$ contain at most 1 such node. By the definition of ρ, the maximum number of nodes N in a ball of radius R is therefore bounded by

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

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

Remark 1:
Theorem 5.22 even holds if the nodes only know the distances up to a constant factor or if the network graph is a doubling quasi unit ball graph instead of a UBG. It is not hard to see that all results of this section remain true up to constant factors in these cases. Let $d(u,v)$ be the real distance between u and v and let $\widehat{d}(u,v)$ be the distance as it is measured by u and v. Further assume that we have $d(u,v)/c \leq \widehat{d}(u,v) \leq cd(u,v)$ for some constant c. With respect to the known distances, every ball of radius $cd(u,v)$ can then be covered by 2^{ρ} balls of radius $\widehat{d}(u,v)/(2c)$. Since we do not make use of the triangle inequality in any part of the analysis, all results still hold if every occurrence of ρ is substituted by $\rho' = \rho\log(c^2)$.

Remark 2:
The results of this section can be extended to other situations than the unit ball graph model. 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 the ball $B_r(x)$ for some radius r. Theorem 5.22 shows that choosing the radius $r \in \mathrm{O}(\log^* n)$ suffices for 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 has a point in S at distance less than ε. In algorithms for metric spaces, ε-nets are a widely used structure [62]. Theorem 5.22 shows that every node can decide whether it is in S based on its $\mathrm{O}(\varepsilon\log^* n)$-neighborhood only.

5.5 Local Approximation Schemes

As described in Section 5.2, for $K_{1,t}$-free graphs, an MIS is a t-approximation for the minimum dominating set and maximum independent set problems. Consequently, an MIS is a constant-factor approximation for the two problems on growth-bounded graphs. The MIS and decomposition algorithms of Sections 5.3 and 5.4 can therefore be used to compute constant minimum dominating set and maximum independent set approximations for growth-bounded graphs. This is significantly stronger than what we can achieve for general graphs. However, none of the algorithms is capable of computing an arbitrarily good approximation, that is, a $(1+\varepsilon)$-approximation for any $\varepsilon > 0$.

At least for UDGs, this is in contrast to the centralized case, in which polynomial time approximation schemes (PTAS) for independent set and dominating set problems in geometric intersection graphs, especially in disk graphs, have been studied in great detail. Most approaches that yield a PTAS exploit the geometric graph representation, and then apply a shifting strategy introduced in [14, 66]. In [41], a PTAS for the maximum independent set problem on general disk graphs is described, and in [67], the MDS problem is considered for unit disk graphs together with other problems for which the shifting strategy is viable. In [28], a PTAS for the minimum connected dominating set problem is given. Can we exploit those ideas and find a distributed approximation scheme for problems like minimum dominating set?

Approaches based on the shifting strategy are inherently central and cannot efficiently be adapted to work in a distributed context. 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.

Even when considering centralized approaches, the case when there is no geometric representation is significantly different: For unit disk graphs, computing a corresponding representation is an NP-hard problem [24]; in fact, it does not even admit a PTAS [80]. In [107], a PTAS for maximum independent set on unit disk graphs without a given representation has been given. The algorithm described in [107] only relies on the topology of the given graph and does not make use of any coordinate or distance information. The ideas of [107] have been extended to the MDS problem in [106]. In the following, we show that the techniques of [106, 107] can be applied to obtain distributed approximation schemes for all polynomially growth-bounded graphs. Section 5.5.1 outlines the main ideas behind the approximation schemes of [106, 107] and Section 5.5.2 shows how the algorithms can be applied in a distributed setting.

5.5.1 A PTAS for Polynomially Growth-Bounded Graphs

The approximation schemes [106, 107] for minimum dominating set and maximum independent set both work in the same general manner. The algorithms work in phases. In each phase, an arbitrary node v of $G = (V, E)$ is selected. Node v then computes an optimal solution for the respective problem for some bounded neighborhood. The phase is completed by removing the considered local neighborhood from the graph G. In detail, the algorithms are constructed such that in the end, that is, when no nodes are left, a feasible solution is obtained. Further, the neighborhoods are chosen such that the computed solution is a $(1 + \varepsilon)$-approximation for an arbitrary parameter $\varepsilon > 0$.

Let us now have a closer look at the described algorithms. We first consider the maximum independent set case. Let $I_r(v)$ be a maximum independent set of the subgraph of G induced by the r-neighborhood $\Gamma_r^+(v)$ of v. For a given $\varepsilon > 0$, we choose r such that

$$|I_{r+1}(v)| \;\leq\; (1+\varepsilon)|I_r(v)|. \tag{5.2}$$

Let $r(v)$ be the smallest r for which the above criterion holds. The following lemma shows that $r(v) \in \mathrm{O}(1)$ for polynomially growth-bounded graphs.

Lemma 5.23. *Let $G = (V, E)$ be a $p(r)$-growth-bounded graph where $p(r)$ is a polynomial in r. There exists a constant $c = c(\varepsilon)$ such that $r(v) \leq c$.*

Proof. By the definition of $p(r)$-growth-boundedness, we have $|I_r(v)| \leq p(r)$. From the definition of $r(v)$, we have $|I_r(v)| > (1+\varepsilon)|I_{r-1}(v)|$ as long as $r \leq r(v)$. For $r \leq r(v)$, we therefore get

$$p(r) \;\geq\; |I_r(v)| \;>\; (1+\varepsilon)|I_{r-1}(v)| \;>\; \cdots \;>\; (1+\varepsilon)^r|I_0(v)| \;=\; (1+\varepsilon)^r,$$

and the claim follows for every constant ε. □

The maximum independent set algorithm now works as follows. Initially, the independent set I of G is empty. As described, in each phase a node v of G is chosen. The radius $r(v)$ is found by computing $I_r(v)$ until Condition (5.2) is satisfied. Note that since $r(v) \in \mathrm{O}(1)$ and because G is growth-bounded, we have $|I_r(v)| \in \mathrm{O}(1)$ for $r \leq r(v) + 1$. Therefore, $I_r(v)$ can be computed in polynomial time. We add the independent set $I_{r(v)}(v)$ to I and remove all nodes $w \in \Gamma_+(u)$ for $u \in I_{r(v)}(v)$ from G. This procedure is iterated as long as there are nodes in G. Because we remove all neighbors of u from G whenever we add a node u to I, I always remains an independent set of G. It remains to be proved that I is at most by a factor of $(1+\varepsilon)$ smaller than a maximum independent set.

Theorem 5.24. *Let I^* be a maximum independent set of G. For the independent set I computed by the above algorithm, we have $|I^*| \leq (1+\varepsilon)|I|$.*

Proof. Let v_i be the node which computes a local independent set $I_{r(v_i)}$ in phase i. Further let T denote the total number of phases. By the construction of the algorithm, we have

$$\bigcup_{i=1}^{T} \Gamma^+_{r(v_i)+1}(v_i) \;=\; V.$$

We therefore obtain

$$(1+\varepsilon)|I| \;=\; (1+\varepsilon)\sum_{i=1}^{T}|I_{r(v_i)}(v_i)| \;\geq\; \sum_{i=1}^{T}|I_{r(v_i)+1}(v_i)| \;\geq\; |I^*|.$$

□

For the minimum dominating set problem, we proceed in an analogous fashion. Let $D_r(v)$ be a minimum cardinality subset of the nodes of G such that all nodes of $\Gamma_r^+(v)$ are covered. Let $r(v)$ denote the minimum radius r for which

$$|D_{r+2}(v)| \leq (1+\varepsilon)|D_r(v)|. \tag{5.3}$$

As in the maximum independent set case, $r(v)$ can be bounded by a constant for polynomially growth-bounded graphs.

Lemma 5.25. *Let $G = (V, E)$ be a $p(r)$-growth-bounded graph where $p(r)$ is a polynomial in r. There exists a constant $c = c(\varepsilon)$ such that $r(v) \leq c$.*

Proof. We use the fact that any maximal independent set is also a dominating set. We therefore have $|I_r(v)| \geq |D_r(v)|$. If r is even and $r \leq r(v)$, we obtain

$$p(r) \geq |I_r(v)| \geq |D_r(v)| > (1+\varepsilon)|I_{r-2}(v)| > \cdots > (1+\varepsilon)^{r/2}|D_0(v)| = (1+\varepsilon)^{r/2}$$

in analogy to Lemma 5.23. For odd r, a similar argumentation holds and thus the claim follows for every constant ε. □

As in the independent set algorithm, in each phase a node v is selected. Initially the dominating set D is empty. Node v adds $D_{r(v)+2}(v)$ to D and removes all nodes from G which are covered by nodes in $D_{r(v)+2}$. Because only covered nodes are removed from G, D is a dominating set at the end of the algorithm. The following theorem shows that D is only by a factor of $(1+\varepsilon)$ larger than an optimal dominating set.

Theorem 5.26. *Let D^* be a minimum dominating set of G and let D be the dominating set computed by the described algorithm. We have $|D| \leq (1+\varepsilon)|D^*|$.*

Proof. As in the proof of Theorem 5.24, let v_i be the node computing a local solution in phase i. After adding $D_{r(v_i)+2}(v_i)$ to D, all nodes which are covered by $D_{r(v_i)+2}(v_i)$ are removed from G. In particular, by the definition of $D_{r(v_i)+2}(v_i)$, all nodes in $\Gamma^+_{r(v_i)+2}(v)$ are removed from G. If T denotes the total number of phases, we therefore have

$$|D| = \left|\bigcup_{i=1}^{T} D_{r(v_i)+2}(v)\right| \leq \sum_{i=1}^{T} |D_{r(v_i)+2}(v)| \leq (1+\varepsilon)\sum_{i=1}^{T} |D_{r(v_i)}(v)| \leq (1+\varepsilon)|D^*|.$$

□

5.5.2 Distributed Approximation Schemes

The approximation schemes described in the previous section seem to be perfectly suited for a distributed implementation. The final solutions are composed of many small (constant) local solutions. The only problem is that different local solutions have to be selected such that they do not intersect too much.

Algorithm 12 Local approximation scheme for independent and dominating sets

```
1: compute MIS S of G;
2: make cluster C_u for each u ∈ S;
3: each v ∉ S joins a cluster C_u for u ∈ Γ(v);
4: color clusters such that clusters containing nodes at distance at most 2R+7
   get different colors;
5: for all cluster colors x do
6:   while ∃v ∈ C_u where color(C_u) = x do
7:     select a node v ∈ C_u for each cluster C_u with color(C_u) = x;
8:     v computes local solution according to Section 5.5.1
9:   od
10: od
```

Let R be the largest possible radius $r(v)$. According to the proofs of Lemmas 5.23 and 5.25, on polynomially growth-bounded graphs, for both problems, there is a constant c such that R is

$$R \;=\; \min_r\{r^c \geq (1+\varepsilon)^r\} \;\in\; \mathrm{O}\left(\frac{\log(1/\varepsilon)}{\varepsilon}\right).$$

For the maximum independent set algorithm, two local solutions $I_{r(v_i)}$ and $I_{r(v_j)}$ do not interfere if $d_G(v_i,v_j) \geq r(v_i)+r(v_j)+1$. For the MDS algorithm, two local solutions $D_{r(v_i)+2}$ and $D_{r(v_j)+2}$ do not interfere if $d_G(v_i,v_j) \geq r(v_i)+r(v_j)+7$. Hence, if u and v are chosen such that $d_G(u,v) \geq 2R+7$, local computations by u and v do not interfere in both cases. They can therefore be carried out in parallel. For each phase of the algorithm, the goal is to find a large set of nodes which have pairwise distance at least $2R+7$. All nodes v of this set can then compute the local solution $I_{r(v)}(v)$ or $D_{r(v)+2}(v)$ in parallel. In detail, the distributed algorithm is given by Algorithm 12. The following theorem shows that for any constant $\varepsilon > 0$, the algorithm computes $(1+\varepsilon)$-approximations for maximum independent set or minimum dominating set in time $\mathrm{O}(T_{\mathrm{MIS}}+\log^* n)$ where T_{MIS} denotes the time to compute an MIS of G.

Theorem 5.27. *In the $\mathcal{LOCAL}$ model, Algorithm 12 computes $(1+\varepsilon)$-approximations for maximum independent set or minimum dominating set in time*

$$\mathrm{O}\left(T_{\mathrm{MIS}} + \left(\frac{\log(1/\varepsilon)}{\varepsilon}\right)\log^* n + \left(\frac{\log(1/\varepsilon)}{\varepsilon}\right)^{\mathrm{O}(1)}\right)$$

on polynomially growth-bounded graphs.

Proof. The algorithm is constructed such that no two nodes at distance less than $2R+7$ compute local independent or dominating sets at the same time. The result of the algorithm therefore is the same as for a sequential execution

as described in Section 5.5.1. It therefore follows that an approximation ratio of $(1+\varepsilon)$ is achieved.

Let us now consider the time complexity of Algorithm 12. The first three steps are clearly within the given bounds. Let $\mathcal{G}$ be the graph for which we compute a coloring in Line 4. The nodes of $\mathcal{G}$ are all nodes of the MIS S, two nodes $u, v \in S$ are connected if and only if their distance with respect to the network graph G is at most $2R+7$. Because $R \in \mathrm{O}(\log(1/\varepsilon)/\varepsilon)$ and G is polynomially growth-bounded, the degree of $\mathcal{G}$ is at most $\mathrm{O}(R^{\mathrm{O}(1)})$. Sending a message on $\mathcal{G}$ needs $\mathrm{O}(R)$ time. Therefore, $\mathcal{G}$ can be colored in $\mathrm{O}(R \log^* n + R^{\mathrm{O}(1)})$ rounds (cf. Chapter 4).

The two loops in Lines 5–10 remain to be looked at. Because $\mathcal{G}$ is colored with $\mathrm{O}(R^{\mathrm{O}(1)})$ colors, the for-loop is executed $\mathrm{O}(R^{\mathrm{O}(1)})$ times. Computing a local solution in Lines 7 and 8 can be done in $\mathrm{O}(R)$ rounds by the definition of R. If we can show that the while-loop is only executed a constant number of times for each color x, the theorem follows. To prove this, we need to have a closer look at the local dominating or independent sets computed in Lines 7 and 8. In the maximum independent set case, all nodes in $I_{r(v)}(v)$ and all neighbors of nodes in $I_{r(v)}(v)$ are removed from G. In the MDS case, all nodes covered by nodes from $D_{r(v)+2}(v)$ are removed from G. In both cases, this includes all nodes of $\Gamma^+(v)$. Assume that G is f-growth-bounded. For each color x, after at most $f(1) \in \mathrm{O}(1)$ iterations of the while-loop, all nodes of a cluster C_u with color x are removed from G. This completes the proof. □

Combined with Theorems 5.13 and 5.22, we obtain the following two corollaries.

Corollary 5.28. *Let G be a polynomially growth-bounded graph. For any constant $\varepsilon > 0$, $(1+\varepsilon)$-approximations for minimum dominating set and maximum matching can be computed in $\mathrm{O}(\log \Delta \log^* n)$ rounds in the $\mathcal{LOCAL}$ model.*

Corollary 5.29. *Let G be a doubling unit ball graph. If nodes know the distances to their neighbors, for any constant $\varepsilon > 0$, $(1+\varepsilon)$-approximations for minimum dominating set and maximum matching can be computed in $\mathrm{O}(\log^* n)$ rounds in the $\mathcal{LOCAL}$ model.*

Chapter 6

Conclusions and Outlook

When devising algorithms which run on computer networks, we encounter difficulties that are not present in standard non-distributed settings. In most cases, the input for a problem at hand is distributed among the nodes of the network and nodes need to communicate with each other in order to obtain the data from other nodes. In a large network, it is in general not possible to learn the complete input. Thus, each node has to compute its part of the output based on partial information about the state of the system. We saw that most importantly, in k rounds, every node can only gather data from its k-hop neighborhood in the network graph. Hence, all nodes have to base their decisions on a local view. We used the $\mathcal{LOCAL}$ model to study this locality condition and the following associated question: How good can a global solution be if the output of every node has to be based on local information?

We analyzed the price of locality for different problems and different classes of network graphs. For distributed covering and packing problems, strong upper and lower bounds on the possible trade-offs between time complexity (locality) and achievable approximation ratio in the $\mathcal{LOCAL}$ model were found. For distributed graph coloring, we established a new lower bound for one-round algorithms which almost matches the best known upper bound. All these results are based on the assumption that the network topology can be arbitrary. Restricting the class of possible network graphs can lead to much faster algorithms. We introduced the class of growth-bounded graphs which we believe to be a reasonable graph model for wireless ad hoc and sensor networks. For growth-bounded graphs and certain natural sub-classes of these graphs, many important distributed problems become significantly easier.

In reality, locality is not the only factor restricting the information upon which nodes have to base their computations. Learning the complete k-hop neighborhood in k rounds can in general require unreasonably large messages. Therefore, bandwidth restrictions further reduce the amount of information which can be gathered in k communication rounds. To comply with this fact, we introduced the $\mathcal{CONGEST}$ and $\mathcal{CONGEST}_{\mathrm{BC}}$ models. Many of the presented

algorithms are designed to also work in these much more restricting models. In particular, we showed that for covering and packing problems almost the same trade-offs between time complexity k and approximation ratio can be achieved in the $\mathcal{LOCAL}$ model and in the $\mathcal{CONGEST}_{\mathrm{BC}}$ model. Note that this is only true as long as the diameter of the network is large. If the diameter is at most k, problems can be solved optimally in the $\mathcal{LOCAL}$ model. In the $\mathcal{CONGEST}$ and $\mathcal{CONGEST}_{\mathrm{BC}}$ models however, we do not know how to improve the trade-off between time complexity and approximation ratio even for constant diameter graphs. Studying the complexity of distributed problems in the $\mathcal{CONGEST}$ and $\mathcal{CONGEST}_{\mathrm{BC}}$ models for constant-diameter graphs is a fascinating open problem.

The presence of large, complex, and sometimes even dynamic computer networks such as the Internet, peer-to-peer networks, or wireless ad hoc and sensor networks shows that it is a necessity to understand the complexity of distributed algorithms. This thesis does not provide a general theory for local computations. It does however introduce tools and techniques which we believe are important steps towards developing such a theory. We can of course not yet know the exact significance of our results for future research.

The techniques used to solve fractional covering and packing problems can be adapted for other types of linear programs. In fact, it seems that all set-cover-like greedy algorithms can be converted into a distributed protocol for the respective fractional problems using the method described in Chapter 2 (see e.g. [101]). It is not clear whether the method can still be applied if we consider even more general LPs. However, studying the fractional versions of distributed problems seems to be a good idea in general, because it allows to avoid the symmetry breaking problem. Based on the current knowledge, the lower bound results of Chapter 3 appear to be the most significant ones of this thesis. It will be interesting to see whether the used techniques will find additional applications in the future. Our work on distributed coloring can be seen as one step on a long way towards understanding one of the most important problems in the area of distributed computing. Although understanding the one-round case is an important step, new insight is needed to generalize the obtained results to two and more rounds. Chapter 5, which considers growth-bounded network graphs, shows one of the directions which future research in the area should follow. While it is definitely important to study the general graph case, it is also fundamental to study network topologies which really occur in practice.

In this thesis, we have made a step forward in an extremely fascinating area which has been started 18 years ago. During these 18 years, only slow progress has been made regarding the core problem of the area, namely understanding the distributed complexity of problems in the $\mathcal{LOCAL}$ model. We hope that this thesis brings new life into the intriguing research on locality phenomena in graphs in general and in distributed computations in particular.

Bibliography

[1] 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.

[2] 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.

[3] K. Alzoubi, P.-J. Wan, and O. Frieder. Message-optimal connected dominating sets in mobile ad hoc networks. In *Proceedings of the 3rd ACM Int. Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, pages 157–164, 2002.

[4] P. Assouad. Plongements lipschitziens dans $\mathbf{R}^n$. *Bull. Soc. Math. France*, 111(4):429–448, 1983.

[5] B. Awerbuch. Complexity of network synchronization. *Journal of the ACM*, 32(4):804–823, 1985.

[6] B. Awerbuch. Optimal distributed algorithms for minimum-weight spanning tree, counting, leader election and related problems. In *Proc. of the 19th Symposium on Theory of Computing (STOC)*, pages 230–240, 1987.

[7] 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.

[8] 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.

[9] B. Awerbuch, A. V. Goldberg, M. Luby, and S. A. Plotkin. Network decomposition and locality in distributed computation. In *Proc. of the 30th Symposium on Foundations of Computer Science (FOCS)*, pages 364–369, 1989.

[10] B. Awerbuch and D. Peleg. Network synchronization with polylogarithmic overhead. In *Proc. of the 31st Symposium on Foundations of Computer Science (FOCS)*, pages 514–522, 1990.

[11] B. Awerbuch and D. Peleg. Sparse partitions. In *Proc. of the 31st Symposium on Foundations of Computer Science (FOCS)*, pages 503–513, 1990.

[12] B. Awerbuch and D. Peleg. Routing with polynomial communication-space trade-off. *SIAM Journal on Mathematics*, 5(2):151–162, 1992.

[13] B. Awerbuch and D. Peleg. Online tracking of mobile users. *Journal of the ACM*, 42(5):1021–1058, 1995.

[14] B. Baker. Approximation algorithms for NP-complete problems on planar graphs. *Journal of the ACM*, 41(1):153–180, 1994.

[15] D. J. Baker and A. Ephremides. The architectural organization of a mobile radio network via a distributed algorithm. *IEEE Trans. Communications*, COM-29(11):1694–1701, 1981.

[16] L. Barrière, P. Fraigniaud, and L. Narayanan. Robust position-based routing in wireless ad hoc networks with unstable transmission ranges. In *Proc. of the 5th Int. Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIALM)*, pages 19–27, 2001.

[17] Y. Bartal, J. W. Byers, and D. Raz. Global optimization using local information with applications to flow control. In *Proc. of the 38th IEEE Symposium on the Foundations of Computer Science (FOCS)*, pages 303–312, 1997.

[18] S. Basagni. A distributed algorithm for finding a maximal weighted independent set in wireless networks. In *Proc. of the 11th IASTED Int. Conference on Parallel and Distributed Computing and Systems (PDCS)*, pages 517–522, 1999.

[19] S. Basagni. Distributed clustering for ad hoc networks. In *Proc. of the IEEE Int. Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN)*, pages 310–315, 1999.

[20] M. Bellare, O. Goldreich, and M. Sudan. Free bits, PCPs and non-approximability – towards tight results. *SIAM Journal on Computing*, 27:804–915, 1998.

[21] R. Bischoff and R. Wattenhofer. Analyzing connectivity-based, multi-hop ad-hoc positioning. In *Proc. of 2nd IEEE Int. Conference on Pervasive Computing and Communications (PERCOM)*, pages 165–176, 2004.

[22] B. Bollobás. *Extremal Graph Theory.* Academic Press, New York, 1978.

[23] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. In *Proc. of Discrete Algorithms and Methods for Mobility (DIALM)*, pages 48–55, 1999.

[24] H. Breu and D. G. Kirkpatrick. Unit disk graph recognition is np-hard. *Computational Geometry. Theory and Applications*, 9(1-2):3–24, 1998.

[25] S. Capkun, M. Hamdi, and J. P. Hubaux. GPS-free positioning in mobile ad-hoc networks. *Cluster Computing*, 5(2), 2002.

[26] M. Cardei, X. Cheng, X. Cheng, and D. Z. Du. Connected domination in multihop ad hoc wireless networks. In *Proceedings of the 6th Int. Conference on Computer Science and Informatics*, 2002.

[27] M. Chatterjee, S. K. Das, and D. Turgut. An on-demand weighted clustering algorithm (WCA) for ad-hoc networks. In *Proc. of IEEE GLOBECOM*, pages 1697–1701, 2000.

[28] 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.

[29] C. Chiang, H. Wu, W. Liu, and M. Gerla. Routing in clustered multihop, mobile wireless networks. In *Proc. of the IEEE Singapore International Conference on Networks*, pages 197–211, 1997.

[30] F. Chin and H. F. Ting. An almost linear time and $O(n \log(n) + e)$ messages distributed algorithm for minimum-weight spanning trees. In *Proc. of the 26th Symposium on Foundations of Computer Science (FOCS)*, pages 257–266, 1985.

[31] V. Chvátal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3), 1979.

[32] V. Chvátal. *Linear Programming*. W. H. Freeman and Company, 1983.

[33] R. Cole and U. Vishkin. Deterministic coin tossing with applications to optimal parallel list ranking. *Information and Control*, 70(1):32–53, 1986.

[34] A. Czygrinow, M. Hańćkowiak, and M. Karoński. Distributed $O(\delta \log n)$-edge-coloring algorithm. In *Proc. of 9th Annual European Symposium on Algorithms (ESA)*, volume 2161 of *LNCS*, pages 345–355, 2001.

[35] G. De Marco and A. Pelc. Fast distributed graph coloring with $O(\Delta)$ colors. In *Proc. of the 12th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 630–635, 2001.

[36] D. Dubhashi, A. Mei, A. Panconesi, J. Radhakrishnan, and A. Srinivasan. Fast distributed algorithms for (weakly) connected dominating sets and linear-size skeletons. In *Proc. of the 14th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 717–724, 2003.

[37] 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 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 517–522, 2003.

[38] M. Elkin. A faster distributed protocol for constructing a minimum spanning tree. In *Proc. of the 15th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 359–368, 2004.

[39] M. Elkin. Unconditional lower bounds on the time-approximation tradeoffs for the distributed minimum spanning tree problem. In *Proc. of the 36th ACM Symposium on Theory of Computing (STOC)*, pages 331–340, 2004.

[40] P. Erdős, P. Frankl, and Z. Füredi. Families of finite sets in which no set is covered by the union of r others. *Israel Journal of Mathematics*, 51:79–89, 1985.

[41] T. Erlebach, K. Jansen, and E. Seidel. Polynomial-time approximation schemes for geometric graphs. In *Proc. of the 12th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 671–679, 2001.

[42] U. Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.

[43] U. Feige and J. Kilian. Zero knowledge and the chromatic number. *Journal of Computer and System Sciences*, 57:187–199, 1998.

[44] F. Fich and E. Ruppert. Hundreds of impossibility results for distributed computing. *Distributed Computing*, 16(2-3):121–163, 2003.

[45] 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.

[46] L. Fleischer. Approximating fractional multicommodity flow independent of the number of commodities. *SIAM Journal on Discrete Math.*, 13(4):505–520, 2000.

[47] L. Fleischer. A fast approximation scheme for fractional covering problems with variable upper bounds. In *Proc. of the 15th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2004.

[48] S. Funke, A. Kesselmann, Z. Lotker, and M. Segal. Improved approximation algorithms for connected sensor cover. In *Proc. of 3rd International Conference on AD-HOC Networks & Wireless (ADHOC-NOW)*, 2004.

[49] E. Gafni. Improvements in the time complexity of two message-optimal election algorithms. In *Proc. of the 4th Symposium on Principles of Distributed Computing (PODC)*, pages 175–185, 1985.

[50] R. G. Gallager, P. A. Humblet, and P. M. Spira. A distributed algorithm for minimum-weight spanning trees. *ACM Trans. on Programming Languages and Systems*, 5:66–77, 1983.

[51] J. Gao, L. Guibas, J. Hershberger, L. Zhang, and A. Zhu. Discrete mobile centers. In *Proc. of the 17th Annual Symposium on Computational Geometry (SCG)*, pages 188–196, 2001.

[52] 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.

[53] M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[54] M. Gerla and J. Tsai. Multicluster, mobile, multimedia radio network. *ACM/Baltzer Journal of Wireless Networks*, 1(3):255–265, 1995.

[55] A. Goldberg, S. Plotkin, and G. Shannon. Parallel symmetry-breaking in sparse graphs. *SIAM Journal on Discrete Mathematics*, 1(4):434–446, 1988.

[56] D. A. Grable and A. Panconesi. Nearly optimal distributed edge coloring in $O(\log \log n)$ rounds. *Random Structures and Algorithms*, 10(3):385–405, 1997.

[57] F. Grandoni, J. Könemann, A. Panconesi, and M. Sozio. Primal-dual based distributed algorithms for vertex cover with semi-hard capacities. In *Proc. of the 24th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 118–125, 2005.

[58] S. Guha and S. Khuller. Approximation algorithms for connected dominating sets. In *Proc. of the 4th Annual European Symposium on Algorithms (ESA)*, volume 1136 of *Lecture Notes in Computer Science*, pages 179–193, 1996.

[59] 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)*, 2003.

[60] M. Hańćkoviak, M. Karoński, and A. Panconesi. On the distributed complexity of computing maximal matchings. In *Proc. of the 9th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 219–225, 1998.

[61] M. Hańćkoviak, M. Karoński, and A. Panconesi. A faster distributed algorithm for computing maximal matchings deterministically. In *Proc. of the 18th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 219–228, 1999.

[62] J. Heinonen. *Lectures on Analysis of Metric Spaces*. Springer-Verlag, 2001. New York.

[63] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proc. of the 33rd Annual Hawaii Int. Conference on System Sciences*, pages 3005–3014, 2000.

[64] T. Hermann and S. Tixeuil. A distributed TDMA slot assignment algorithm for wireless sensor networks. In *Proc. of 1st Workshop on Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS)*, volume 3121 of *Lecture Notes in Computer Science*, pages 45–58, 2004.

[65] D. Hochbaum, editor. *Approximation Algorithms for NP-hard Problems*. PWS Publishing Company, 1997.

[66] D. Hochbaum and W. Maass. Approximation schemes for covering and packing problems. *Journal of the ACM*, 32(1):130–136, 1985.

[67] 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. *Journal of Algorithms*, 26(2):238–274, 1998.

[68] A. Israeli and A. Itai. A fast and simple randomized parallel algorithm for maximal matching. *Information Processing Letters*, 22:77–80, 1986.

[69] L. Jia, R. Rajaraman, and R. Suel. An efficient distributed algorithm for constructing small dominating sets. *Distributed Computing*, 15(4):193–205, 2002.

[70] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, volume 353, chapter 5. 1996.

[71] D. S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9:256–278, 1974.

[72] R. E. Johnson and F. B. Schneider. Symmetry and similarity in distributed systems. In *Proc. of the 4th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 13–22, 1985.

[73] R. M. Karp. Reducibility among combinatorial problems. In *Proc. of a Symposium on the Complexity of Computer Computations*, pages 85–103, 1972.

[74] J. Kleinberg, A. Slivkins, and T. Wexler. Triangulation and embedding using small sets of beacons. In *Proc. of the 45th IEEE Symposium on Foundations of Computer Science (FOCS)*, 2004.

[75] S. G. Kolliopoulos and N. E. Young. Tight approximation results for general covering integer programs. In *Proc. of the 42nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 522–528, 2001.

[76] R. Krauthgamer and J. Lee. Navigating nets: Simple algorithms for proximity search. In *Proc. of the 15th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, 2004.

[77] P. Krishna, M. Chatterjee, N. H. Vaidya, and D. K. Pradhan. A cluster-based approach for routing in ad-hoc networks. In *Proc. of the 2nd USENIX Symposium on Mobile and Location-Independent Computing*, pages 1–10, 1995.

[78] F. Kuhn, T. Moscibroda, and R. Wattenhofer. Initializing newly deployed ad hoc and sensor networks. In *Proc. of the 10th Annual Int. Conference on Mobile Computing and Networking (MOBICOM)*, pages 260–274, 2004.

[79] F. Kuhn, T. Moscibroda, and R. Wattenhofer. Radio clustering from scratch. In *Proc. of 12th Annual European Symposium on Algorithms (ESA)*, volume 3221 of *LNCS*, pages 460–471, 2004.

[80] F. Kuhn, T. Moscibroda, and R. Wattenhofer. Unit disk graph approximation. In *Proc. of the 2nd Joint Workshop on Foundations of Mobile Computing (DIALM-POMC)*, pages 17–23, 2004.

[81] F. Kuhn and R. Wattenhofer. Constant-time distributed dominating set approximation. In *Proc. of the 22nd ACM Symposium on Principles of Distributed Computing (PODC)*, pages 25–32, 2003.

[82] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger. Geometric ad-hoc routing: Of theory and practice. In *Proc. of the 22nd ACM Symposium on Principles of Distributed Computing (PODC)*, pages 63–72, 2003.

[83] F. Kuhn, R. Wattenhofer, and A. Zollinger. Ad-hoc networks beyong unit disk graphs. In *Proc. of the 1st Joint Workshop on Foundations of Mobile Computing (DIALM-POMC)*, pages 69–78, 2003.

[84] F. Kuhn, R. Wattenhofer, and A. Zollinger. Worst-case optimal and average-case efficient geometric ad-hoc routing. In *Proc. of the 4th ACM Int. Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, pages 267–278, 2003.

[85] E. Kushilevitz and Y. Mansour. An $\omega(d \log(n/d))$ lower bound for broadcast in radio networks. *SIAM Journal on Computing*, 27(3):702–712, 1998.

[86] S. Kutten and D. Peleg. Fast distributed construction of small k-dominating sets and applications. *Journal of Algorithms*, 28:40–66, 1998.

[87] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.

[88] 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):275–284, 1995.

[89] 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.

[90] C. R. Lin and M. Gerla. Adaptive clustering in mobile wireless networks. *IEEE Journal on Selected Areas in Communications*, 16:1265–1275, 1997.

[91] N. Linial. Locality in distributed graph algorithms. *SIAM Journal on Computing*, 21(1):193–201, February 1992.

[92] N. Linial. Local-global phenomena in graphs. *Combinatorics Probability and Computing*, 2:491–503, 1993.

[93] N. Linial and M. Saks. Low diameter graph decompositions. *Combinatorica*, 13(4):441–454, 1993.

[94] Z. Lotker, B. Patt-Shamir, and D. Peleg. Distributed MST for constant diameter graphs. In *Proc. of the 20th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 63–71, 2001.

[95] Z. Lotker, E. Pavlov, B. Patt-Shamir, and D. Peleg. MST construction in $O(\log \log n)$ communication rounds. In *Proc. of the 16th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 94–100, 2003.

[96] M. Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM Journal on Computing*, 15:1036–1053, 1986.

[97] 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.

[98] 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.

[99] T. Moscibroda and R. Wattenhofer. Efficient computation of maximal independent sets in unstructured multi-hop radio networks. In *Proc. of 1st IEEE Int. Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, pages 51–59, 2004.

[100] T. Moscibroda and R. Wattenhofer. Coloring unstructured radio networks. In *Proc. of the 17th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 39–48, 2005.

[101] T. Moscibroda and R. Wattenhofer. Facility location: Distributed approximation. In *Proc. of the 24th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 108–117, 2005.

[102] T. Moscibroda and R. Wattenhofer. Maximal independent sets in radio networks. In *Proc. of the 24th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 148–157, 2005.

[103] T. Moscibroda and R. Wattenhofer. Maximizing the lifetime of dominating sets. In *Proc. of the 5th International Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks (WMAN)*, 2005.

[104] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[105] M. Naor and L. Stockmeyer. What can be computed locally? In *Proc. of the 25th Annual ACM Symposium on Theory of Computing (STOC)*, pages 184–193, 1993.

[106] T. Nieberg and J. Hurink. A PTAS for the minimum dominating set problem in unit disk graphs. Technical Report 1732, Dept. of Applied Mathematics, University of Twente, Enschede, The Netherlands, 2004.

[107] T. Nieberg, J. Hurink, and W. Kern. A robust PTAS for maximum independent sets in unit disk graphs. In *Proc. of the 30th Workshop on Graph Theoretic Concepts in Computer Science*, pages 214–221, 2004.

[108] A. Panconesi and A. Srinivasan. On the complexity of distributed network decomposition. *Journal of Algorithms*, 20(2):581–592, 1995.

[109] A. Panconesi and A. Srinivasan. Randomized distributed edge coloring via an extension of the Chernoff-Hoeffding bounds. *SIAM Journal on Computing*, 26(2):350–368, 1997.

[110] C. Papadimitriou and M. Yannakakis. Linear programming without the matrix. In *Proc. of the 25th ACM Symposium on Theory of Computing (STOC)*, pages 121–129, 1993.

[111] A. Pelc. Personal communication.

[112] D. Peleg. *Distributed Computing: A Locality-Sensitive Approach*. SIAM, 2000.

[113] D. Peleg and V. Rubinovich. A near-tight lower bound on the time complexity of distributed MST construction. *SIAM Journal on Computing*, 30(5):1427–1442, 2001.

[114] C. E. Perkins and E. M. Royer. Ad hoc on-demand distance vector routing. In *Proc. of 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, 1999.

[115] C. G. Plaxton, R. Rajaraman, and A. W. Richa. Accessing nearby copies of replicated objects in a distributed environment. In *Proceedings of the 9th ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 311–320, 1997.

[116] S. Plotkin, D. Shmoys, and E. Tardos. Fast approximation algorithms for fractional packing and covering problems. *Mathematics of Operations Research*, 20:257–301, 1995.

[117] P. Raghavan and C. D. Thompson. Randomized rounding: A technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7(4):365–374, 1987.

[118] 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.

[119] S. Ramanathan. A unified framework and algorithm for channel assignment in wireless networks. *Wireless Networks*, 5:81–94, 1999.

[120] E. M. Royer and C. Toh. A review of current routing protocols for ad-hoc mobile wireless networks. In *IEEE Personal Communications*, volume 6, April 1999.

[121] A. Savvides, C.-C. Han, and M. B. Strivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. In *Proc. of the 7th Int. Conference on Mobile Computing and Networking (MOBICOM)*, pages 166–179, 2001.

[122] P. Sinha, R. Sivakumar, and V. Bharghavan. Enhancing ad hoc routing with dynamic virtual infrastructures. In *Proc. of the 20th IEEE Conference on Computer Communications (INFOCOM)*, pages 1763–1772, 2001.

[123] P. Slavík. A tight analysis of the greedy algorithm for set cover. In *Proc. of the 28th ACM Symposium on Theory of Computing (STOC)*, pages 435–441, 1996.

[124] A. Srinivasan. Improved approximations of packing and covering problems. In *Proc. of the 27th ACM Symposium on Theory of Computing (STOC)*, pages 268–276, 1995.

[125] I. Stojmenovic, M. Seddigh, and J. Zunic. Dominating sets and neighbor elimination-based broadcasting algorithms in wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, 12(12):14–25, 2001.

[126] K. Talwar. Bypassing the embedding: Approximation schemes and compact representations for low dimensional metrics. In *Proc. of the 36th ACM Symposium on Theory of Computing (STOC)*, 2004.

[127] V. Vazirani. *Approximation Algorithms*. Springer, 2001.

[128] Y. Wang and X.-Y. Li. Geometric spanners for wireless ad hoc networks. In *Proc. of the 22nd Int. Conference on Distributed Computing Systems (ICDCS)*, 2002.

[129] Y. Wang and X.-Y. Li. Localized construction of bounded degree and planar spanner for wireless ad hoc networks. In *Proc. of 1st Joint Workshop on Foundations of Mobile Computing (DIALM-POMC)*, pages 59–68, 2003.

[130] M. Wattenhofer and R. Wattenhofer. Distributed weighted matching. In *Proc. of th 18th Int. Conference on Distributed Computing (DISC)*, number 3274 in LNCS, pages 335–348, 2004.

[131] 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 20th INFOCOM*, 2001.

[132] R. Wattenhofer and A. Zollinger. Xtc: A practical topology control algorithm for ad-hoc networks. In *Proc. of the 4th International Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks (WMAN)*, 2004.

[133] J. Wu and H. Li. On calculating connected dominating sets for efficient routing in ad hoc wireless networks. In *Proc. of the 3rd Int. Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIALM)*, pages 7–14, 1999.

[134] N. E. Young. Randomized rounding without solving the linear program. In *Proc. of the 6th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 170–178, 1995.

[135] N. E. Young. Sequential and parallel algorithms for mixed packing and covering. In *Proc. of the 42nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 538–546, 2001.

Curriculum Vitae

July 30, 1976	Born in Basel, Switzerland
1983–1995	Primary school and high schools in Hofstetten-Flüh SO, Bättwil SO, and Oberwil BL, Switzerland
1996	military service
1996–2001	Studies in computer science, ETH Zurich, Switzerland
October 2001	Diploma in computer science, ETH Zurich, Switzerland
2002–2005	Ph.D. student, research and teaching assistent Distributed Computing Group, Prof. Roger Wattenhofer, ETH Zurich, Switzerland
August 2005	Ph.D. degree, Distributed Computing Group, ETH Zurich, Switzerland

Advisor:	Prof. Roger Wattenfhoer
Co-examiner:	Prof. Nathan Linial, Hebrew University, Jerusalem, Israel
Co-examiner:	Prof. Friedhelm Meyer auf der Heide, University of Paderborn, Germany

Publications

In the following, all publications which were written during the three and a half years in which I was a Ph.D. student at ETH Zurich are listed.

1. *The Price of Being Near-Sighted.* Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. 17th ACM-SIAM Symposium on Discrete Algorithms (SODA), Miami, Florida, USA, 2006.

2. *Improved Approximation Algorithms for Connected Sensor Cover.* Stefan Funke, Alexander Kesselman, Fabian Kuhn, Zvi Lotker, and Michael Segal. Wireless Networks Journal.

3. *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.

4. *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.

5. *Interference in Cellular Networks: The Minimum Membership Set Cover Problem.* Fabian Kuhn, Pascal von Rickenbach, Roger Wattenhofer, Emo Welzl, and Aaron Zollinger. 11th International Computing and Combinatorics Conference (COCOON), Kunming, China, August 2005.

6. *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.

7. *A Self-Repairing Peer-to-Peer System Resilient to Dynamic Adversarial Churn.* Fabian Kuhn, Stefan Schmid, and Roger Wattenhofer. 4th International Workshop on Peer-to-Peer Systems (IPTPS), Ithaca, NY, February 2005.

8. *Efficient Adaptive Collect using Randomization.* Hagit Attiya, Fabian Kuhn, Mirjam Wattenhofer, and Roger Wattenhofer. 18th Annual International Conference on Distributed Computing (DISC), Amsterdam, Netherlands, October 2004.

9. *Unit Disk Graph Approximation.* Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. 2nd ACM DIALM-POMC Joint Workshop on Foundations of Mobile Computing (DIALM-POMC), Philadelphia, PA, October 2004.

10. *Initializing Newly Deployed Ad Hoc and Sensor Networks.* Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. 10th ACM International Conference on Mobile Computing and Networking (MOBICOM), Philadelphia, PA, September 2004.

11. *Radio Network Clustering from Scratch.* Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. 12th Annual European Symposium on Algorithms (ESA), Bergen, Norway, September 2004.

12. *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.

13. *Dynamic Analysis of the Arrow Distributed Protocol.* Fabian Kuhn and Roger Wattenhofer. 16th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA), Barcelona, Spain, June 2004.

14. *Ad-Hoc Networks Beyond Unit Disk Graphs.* Fabian Kuhn, Roger Wattenhofer, and Aaron Zollinger. 1st ACM DIALM-POMC Joint Workshop on Foundations of Mobile Computing (DIALM-POMC), San Diego, California, USA, September 2003.

15. *Constant-Time Distributed Dominating Set Approximation.* Fabian Kuhn and Roger Wattenhofer. 22nd ACM Symposium on the Principles of Distributed Computing (PODC), Boston, Massachusetts, USA, July 2003. To appear in Distributed Computing Journal

16. *Geometric Ad-Hoc Routing: Of Theory and Practice.* Fabian Kuhn, Roger Wattenhofer, Yan Zhang, and Aaron Zollinger. 22nd ACM Symposium on the Principles of Distributed Computing (PODC), Boston, Massachusetts, USA, July 2003.

17. *Worst-Case Optimal and Average-Case Efficient Geometric Ad-Hoc Routing.* Fabian Kuhn, Roger Wattenhofer, and Aaron Zollinger. 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC), Annapolis, Maryland, USA, June 2003.

18. *Asymptotically Optimal Geometric Mobile Ad-Hoc Routing.* Fabian Kuhn, Roger Wattenhofer, and Aaron Zollinger. 6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIALM), Atlanta, Georgia, September 2002.