DISS. ETH NO. 18300

Algorithmic Challenges in Wireless Networks

Interference, Energy and Incentives

A dissertation submitted to

ETH ZURICH

for the degree of

Doctor of Sciences

presented by

Yvonne Anne Pignolet

MSc ETH CS, ETH Zürich

born May 31, 1981

citizen of

Bilten GL and Misery-Courtion FR

accepted on the recommendation of

Prof. Dr. Roger Wattenhofer, examiner
Prof. Dr. James Aspnes, co-examiner
Prof. Dr. Subhash Suri, co-examiner

2009

Abstract

Over the past few decades wireless networks have permeated our lives. We use mobile phones, we operate a WLAN at home, our laptops connect to our printers using the Bluetooth protocol, to name but a few examples. This thesis investigates some of the theoretical possibilities and limitations of wireless networks.

The characteristics of wireless communication pose some challenges not present in wired networks. E.g., mutual interference impairs the quality of the signals received and might even prevent the correct reception of messages. Efficient power control and scheduling algorithms that coordinate the transmissions are therefore essential for the operation of wireless networks. Moreover, due to the shared nature of the communication medium, harmful adversarial attacks are easier to implement, e.g., by jamming a frequency band. Thus, algorithms that guarantee communication despite such disruptions are necessary. This thesis addresses these exigent problems and provides lower bounds and algorithms to meet these challenges.

Another difficulty is caused by the fact that wireless devices are typically equipped with a battery as a source of energy. Recharging this battery may be tedious or even impossible. In order to prolong the lifetime of a network, energy-efficient algorithms for wireless networks are needed. We offer answers to the question of how messages can be aggregated with the twofold objective of minimizing delay and energy consumption simultaneously.

Usually, wireless devices of a network are assumed to collaborate on a common application such as environmental monitoring. However, similar to agents in socio-economic systems, the participants of a large network may operate on a decentralized control regime just as often, and represent various stake-holders with conflicting objectives. In many distributed systems, the rules of interaction cannot be changed. However, a system designer may be able to influence the agents' behavior by offering payments for certain outcomes. Thus, a designer faces the following optimization problem: How can a desired outcome be implemented at minimal cost? And to what extent can the social welfare be influenced within the bounds of economic rationality, that is, by taking the implementation cost into account? In this thesis, we aim to lay the computational and algorithmic foundations of a solution to this problem. Besides considering classic, benevolent mechanism designers, this thesis analyzes how malicious mechanism designers can deteriorate the participants' situation to a larger extent than the amount of money invested.

Zusammenfassung

In den letzten Jahrzehnten haben drahtlose Netzwerke markant an Bedeutung gewonnen. Täglich benützen wir Mobiltelefone, betreiben zu Hause ein WLAN und unsere Laptops verbinden sich via Bluetooth mit dem Drucker. In dieser Dissertation untersuchen wir die theoretischen Möglichkeiten und Grenzen von drahtlosen Netzwerken. Die Eigenschaften der Funkkommunikation führen zu Herausforderungen, die in drahtgebundenen Netzwerken nicht existieren. Beispielsweise beeinflussen sich Funksignale gegenseitig, was die Übertragungsqualität mindert und unter Umständen sogar den korrekten Empfang von Nachrichten verhindern kann. Effiziente Algorithmen für die Steuerung der Sendeleistung und die Koordination der Sendezeitpunkte sind darum unentbehrlich für den Betrieb von Funknetzen. Da das Kommunikationsmedium für alle zugänglich ist, sind zudem böswillige Angriffe einfacher auszuführen, zum Beispiel durch Störsender für ein Frequenzband. Aufgrund dieser Tatsache sind Algorithmen, die es erlauben, die Kommunikation dennoch aufrecht zu erhalten, nötig. Diese Arbeit befasst sich mit diesen grundlegenden Problemen und liefert sowohl untere Schranken als auch Algorithmen, die diese Herausforderungen bewältigen.

Eine weitere Schwierigkeit besteht darin, dass drahtlose Geräte häufig mit einer Batterie als Energiequelle ausgestattet sind. Diese aufzuladen oder auszuwechseln kann mühsam oder sogar unmöglich sein. Um die Laufzeit von Netzwerken zu verlängern braucht es Algorithmen, die den Energieverbrauch kontrollieren. Diese Dissertation bietet Antworten auf die Frage, wie Nachrichten zusammengefasst werden können, mit dem Ziel, Energieverbrauch und Verzögerung gleichzeitig zu minimieren.

Meist wird angenommen, dass die Komponenten eines Netzwerkes zusammenarbeiten, um ein gemeinsames Ziel zu erreichen, wie zum Beispiel die Überwachung eines Gebietes. Mindestens so oft jedoch sind die Teilnehmer eines Netzwerks unabhängig und weisen zum Teil sich widersprechende Interessen auf, ähnlich zu Agenten in einem sozio-ökonomischen System. In vielen verteilten Sytemen können die Interaktionsregeln nicht direkt verändert werden. Trotzdem können die Spieler beeinflusst werden, indem eine Vergütung in Aussicht gestellt wird, für den Fall, dass eine gewisse Situation eintrifft. Wir sind also mit den folgenden Optimierungsproblemen konfrontiert: Wie kann eine gewünschte Situation zu möglichst tiefen Kosten herbeigeführt werden? Und in welchem Ausmass kann das soziale Wohl gesteigert werden im Vergleich zu den anfallenden Kosten? Im zweiten Teil dieser Dissertation legen wir die algorithmischen Grundlagen zu einer Lösung dieses Problems. Abgesehen von klassischen, positiven Effekten analysieren wir auch, wie die Situation der Mitspieler mutwillig um einen grösseren als den investierten Betrag verschlechtert werden kann.

Contents

1

Introduction

In this chapter we briefly survey ad hoc and sensor networks, and then discuss the algorithmic challenges they pose. The last section contains a summary of the contributions of this thesis.

1.1 Ad Hoc and Sensor Networks

The evolution of technology has lead to the development of surprisingly small electronic devices. Moreover, the cost of hardware has been steadily decreasing. As a result, an ever growing number of objects have been equipped with radio communication hardware, and a vast quantity of mobile devices are omnipresent in our lives. Most of us carry a mobile phone, our laptops access the internet via a WLAN, and futuristic scenarios envision diverse applications for wireless communication [31, 70]. Many such applications rely on sensor nodes, which are wireless devices capable of sensing physical phenomena. The anticipated applications cover a broad range of domains, such as health care. Sensor nodes attached to a patient's body could be used to measure his or her vital signs [103, 14]. In transport and logistics, the storage conditions of products can be monitored by sensors and goods can be localized in real-time at production sites or distribution centers [59, 40].

These scenarios require devices to form decentralized wireless networks, where each participating node forwards data to other nodes based on the network connectivity. These so-called ad hoc networks are self-organizing, in contrast to managed infrastructure wireless networks with a designated node coordinating the communication of the other nodes. The decentralized and self-organizing nature of ad hoc networks makes them suitable for a variety of applications in which central nodes cannot be relied on. Furthermore, they may improve the scalability of networks compared to that of wireless managed networks, though theoretical [53] and practical [76] limits to the overall capacity of such networks have been identified. Minimal configuration and

quick deployment make ad hoc networks suitable for emergency situations due to natural disasters or military conflicts.

Over the past few decades, a great amount of research has been devoted to networks. As a consequence, efficient algorithms for a variety of networking tasks have been devised. Most of this work addresses fixed wired networks with stationary nodes. Nevertheless, many of the proposed solutions can be adapted for wireless networks. Thus, several fundamental network communication problems have been solved, yet at least as many are waiting to be tackled. More specifically, the wireless nature of today's networks and the increasing heterogeneity of available devices evoke new challenges.

The aim of this thesis is to lay the necessary theoretical foundations by designing appropriate models and by proving upper and lower bounds for some of the problems arising in these networks.

Hardware Components

Let us look more closely at the hardware of the devices that form wireless networks. The components relevant to wireless communication are:

- A **processor**: processes data and controls the functionality of other components of the device
- A **radio communication unit**: responsible for transmitting and decoding radio signals
- **Memory**: stores the necessary data
- An **energy source**: a battery or a small solar cell

In the design of algorithms for networks consisting of a large numbers of such devices, the capabilities and limitations of these components have to be considered.

Processor: Even though the processors used in today's wireless devices are of considerable speed and power despite their small size, they can be a limiting factor when devising new applications or protocols.

Radio communication unit: Usually, the radio communication unit contains a transceiver, a combined radio frequency transmitter and receiver. Its transmission range varies from a few meters to a few hundred meters. The great success of wireless networks is mainly due to the fact that no expensive infrastructure is necessary for their deployment and to the fact that every device can exchange information with any other device in its reception range. However, this is also the most problematic characteristic of wireless networks. Since the communication medium is shared by all participants, interference has to be dealt with. Anyone can disrupt wireless communication by broadcasting a strong signal, making wireless networks vulnerable to

jamming attacks. In addition to adversarial signals impairing the quality of signals received, simultaneous communication attempts by other devices can also cause disturbance and prevent the correct reception of transmissions.

Memory: Technological progress allows us to store a large amount of data in a tiny space. However, depending on the hardware used, memory can be scarce, e.g., in sensor nodes. Hence, there is a demand for algorithms that use memory sparingly.

Energy source: For the devices discussed here the most critical resource is energy. The devices are typically battery operated and it may be tedious or impossible to recharge or exchange these batteries. Consequently, the devices must be designed to be energy-efficient, causing designers to make energy versus speed trade-offs.

Heterogeneous Users

Another issue influencing the performance of wireless networks does not stem from the hardware, but from the heterogeneity of interacting users. We cannot assume a priori that the users controlling the devices are willing to cooperate. Many networks were built for one specific purpose and under the assumption that every participant closely follows pre-defined protocol. Nowadays, devices can typically be used for many purposes and they can be programmed to execute actions different from those initially intended. This fact particularly applies to wireless networks, where each participant can potentially communicate with any other participant. Therefore, in addition to mere technical challenges, a system designer has to take into account sociological and economic aspects as well when designing protocols to maximize the system's performance. Often, the rules of interaction cannot be changed but a systems designer may be able to influence the participants' behavior by offering financial rewards for certain outcomes.

This short and rather general overview has highlighted some of the aspects that have to be considered when devising algorithms for wireless networks. In the next section, we will point out areas where particular effort is necessary, and describe the problems that this thesis addresses.

1.2 Algorithmic Challenges in Wireless Networks

As demonstrated in the previous section, the performance of a wireless network depends on the successful interaction of several components. We identified those characteristics of wireless communication devices that are the source of most problems in wireless networks. Among them are the following three key problems wireless networks suffer from: interference, a limited availability of energy, and selfish participants with conflicting goals. In the following, we will pinpoint the algorithmic challenges they raise in more detail.

Interference

Efficient device discovery is a fundamental problem in wireless networks. Before a node can perform any particular task, it has to explore its vicinity for potential communication partners and resources. While existing protocols are quite fast under normal conditions, the time that lapses until two devices meet is affected by interference. Because the signals are transmitted by (public) radio waves rather than in (protected) cables, deliberate and accidental interference has an extremely detrimental influence on wireless networks. Accidental interference occurs all too often, owing to the shared nature of the frequency bands in which these networks operate. E.g., it is common for a wireless network, or a part of it, to become unusable when a cordless telephone is near a wireless node. Deliberate jamming attacks are not yet as common as accidental interference, but they are certainly straightforward to implement. One only needs to set up a transmitter and ensure that the signals emitted have sufficient power to overwhelm the wireless network's signals. Jamming attacks can severely interfere with the normal operation of wireless networks. Consequently, device discovery algorithms that can cope with jamming attacks are needed.

Once a number of devices have been able to establish a network, interference continues to be problematic. If two nodes close to each other transmit concurrently, the chances are that neither of their signals can be received correctly. Thus, transmissions have to be coordinated to allow multiple users connected to the same physical medium to share its capacity. This goal is typically achieved by dividing the access time or the frequency range into smaller units and assigning them to the users. Frequency Division Multiple Access (FDMA) assigns one or several frequency bands to each device, allowing them to utilize the allocated radio spectrum without interfering with one another. Another approach, Time Division Multiple Access (TDMA), assigns each device certain time slots. The users transmit in rapid succession, all using their own time slot. The performance of wireless networks depends on the availability of algorithms which effectively coordinate the timing and frequency bands of broadcasting nodes. Note that for the algorithmic question it does not matter whether FDMA or TDMA is applied, as the inherent coordination problem remains the same.[1]

In addition to time and frequency allocation, the signal strengths of the transmitting nodes influence the performance of wireless networks. Nodes emitting signals of different power levels can increase the number of simultaneous transmissions. On the downside, a higher power level causes more interference and can prevent nodes in the vicinity from transmitting successfully. Thus, power control constitutes an additional axis of interest.

[1] In the remainder of thesis we focus on scheduling algorithms that assign time slots to wireless devices. However, these algorithms can be applied to the frequency assignment problem as well.

Limited Energy

Wireless devices often have a limited energy supply, since they are powered by batteries or solar cells. Consequently, energy-efficient algorithms are required. The power consumption of the communication subsystem is often orders of magnitudes greater than the amount energy used for data processing [37, 25]. Hence, the easiest way to save energy is to reduce the number of transmissions at the expense of processing speed. Whereas this idea leads to economical algorithms, we usually also want our algorithms to be fast and report results quickly, a goal that can only be reached with a frequent exchange of messages. In other words, we often face a trade-off between speed and energy expenditure. A prototypical example of this trade-off involves a large network of sensor nodes used to identify regions or resources that are experiencing some phenomenon of particular concern (e.g., seismic activity). The sensor nodes observe their environment and send their measurements, in a multi-hop manner, to a given sink. In order to prolong the lifetime of the network, we can try to use as few transmissions as possible. At the same time, the aim is to minimize the time until the sink is informed about the change of the measured values. The number of transmissions can be reduced by delaying the transmission of information about one event and then aggregating it with information about later events. For instance, a sensor node monitoring the seismic activity in a certain location may delay the transmission of the latest measurements in order to combine these with future measurements. However, it is typically desirable that these notifications are not delayed significantly, but transmitted quickly to a sink node where an observer can, for example, raise an alarm in the case of an irregularity (indicating the location and magnitude of the seismic activity in our example). This aggregation problem needs algorithms that decide whether a node should forward its information towards the sink immediately or wait for additional information for aggregation before transmitting. Thus, if a node knew that all its neighbors' messages would arrive in the next time slot, it would preferably wait for their arrival. However, if the situation were not going to change in the near future, forwarding the information immediatly would reduce the delay. Since these so-called online algorithms cannot foresee the future, they are forced to make their decision based on the current situation. Furthermore, such aggregation algorithms must not be too complex, as the available memory on sensor nodes is very limited.

Cooperation between Heterogeneous and Selfish Users

If users have increased control over their devices, they may be tempted to adjust these in order to maximize their benefit. This selfish behavior can dramatically diminish the efficiency of a network or even paralyze it completely. Game theory is a powerful tool for analyzing decision-making in systems

with autonomous and rational participants. The basic assumption is that the agents are rational and pursue well-defined objectives while taking into account their knowledge or their expectations of the other agents' behavior. Many applications of game theory are related to economics, but it has been applied to numerous fields ranging from law enforcement to voting analysis. Since the participants of a network may not always follow their assigned protocols but may make selfish decisions, game theory provides tools to develop analytical models of node behavior and predict the impact of different protocols and policies on that behavior. Moreover, game theory can help to design systems that offer appropriate incentives for the participants to behave in ways constructive to the network as a whole. A major achievement of game theory is the insight that networks of selfish agents often suffer from inefficiency due to the effects of such behavior. If a game-theoretic analysis of a distributed computing system reveals that this is the case, the protocols of this system should be extended by a mechanism that encourages cooperation. We consider a mechanism designer whose power is to some extent based on her monetary assets, primarily, though, on her creditability. She offers the players monetary incentives for certain outcomes and the players trust her to make the promised payments. When expecting additional payments, rational players will necessarily change their behavior and choose one of the desired outcomes. Thus, a designer faces the following optimization problem: How can a desired outcome be reached at minimal cost? To this end, algorithms assessing the cost of implementing favorable outcomes are necessary.

In view of the above challenges, this thesis analyses the complexity of the problems at hand and provides efficient algorithms accounting for certain characteristics of wireless networks and the traditional objective of scalability. We formulate the main research question this thesis discusses as follows:

How can available resources such as time, frequency channels, energy, and monetary incentives be managed efficiently in wireless networks?

Note that selfish users not only influence the performance of networks, but also every situation in which parties with conflicting objectives interact. In other words, solutions to this problem are applicable to a broad range of situations. Therefore, we decided to treat these aspects in a more general fashion and we divided this thesis into two parts. The first part investigates algorithmic challenges due to the properties of wireless ad hoc and sensor networks, whereas the second part studies how selfish users can be convinced to approach an outcome taking the welfare of all participants into account.

1.3 Thesis Overview

With regard to the previously mentioned challenges, we will describe the main contributions of the thesis in the following.

- **Fast Device Discovery Despite Jamming** We present discovery algorithms with the twofold objective of allowing devices to find one another quickly in the absence of a jammer and to degrade gracefully under adversarial jamming. Let m denote the number of channels the devices can use. We prove, with weak assumptions as to the behavior of the adversary, that there exists an algorithm without knowledge of the jammer's strategy that guarantees that the expected duration until two devices meet is at most a factor of $O(\log^2 m)$ longer than the optimal expected duration when knowing the jammer's strategy. Furthermore, we show that this is the best factor achievable. Our analytical findings are complemented by simulations providing evidence that the algorithms perform well in practice.

- **Uniform Power Scheduling** We study the complexity of finding an optimal schedule for a set of communication requests in the physical signal-to-noise-plus-interference model. We show that such coordination problems are NP-hard in this model and we suggest efficient approximation algorithms to construct a schedule for the users.

- **Power Control and Scheduling** If nodes can adjust their transmission power level, there are communication requests that can be scheduled concurrently, even though a uniform power assignment would have caused too much interference for simultaneous transmission. We define a new measure *disturbance* that captures the intrinsic difficulty of scheduling a set of communication requests. We conduct a worst-case analysis of existing algorithms and demonstrate that low-disturbance scnearios exist under which these algorithms perform poorly; more precisely, they schedule n requests in n time slots. In contrast, we present a new algorithm with provable performance guarantees. Namely, it schedules every set of cardinality n in a number of time slots in $O(\log^2 n)$ times the disturbance of the problem instance.

- **Delay-Sensitive Aggregation** We investigate an aggregation problem in which nodes are organized in a tree topology. We analyze a simple online algorithm balancing the costs of energy and of delay and improve on previous results. More precisely, we derive an upper bound on the competitive ratio of $O(\min(h, c))$, where h is the tree's height, and c is the transmission cost per edge. Moreover, we prove that this upper bound is tight in the sense that any oblivious algorithm has a ratio of at least $\Omega(\min(h, c))$. The best upper bound known so far was

$O(h \log(cn))$, where n is the network size. Furthermore, we prove a tight competitive ratio of $\Theta(\min(\sqrt{h}, c))$ for chain networks, and we introduce a model for online event aggregation, in which the importance of an event depends on its difference from previous events.

- **Mechanism Design by Creditability** We provide the computational and algorithmic foundations for the distribution of monetary incentives. We present algorithms and complexity results for the difficulty of computing the cost of reaching a certain outcome. E.g., we show that we can determine efficiently whether it is possible to influence the performance of a given system merely by creditability, without any payments at all. Besides considering classic, benevolent mechanism designers, we analyze how malicious mechanism designers can corrupt games and negatively affect the players' situation to a larger extent than the amount of money invested. In addition, we suggest alternative rationality models, such as risk-averse behavior, and algorithms to determine the necessary payments for such participants.

Part I

Ad Hoc and Sensor Networks

2

Speed Dating despite Jamming

Many wireless standards and protocols today, such as WLAN and Bluetooth, operate on similar frequency bands. While this permits an efficient usage of the limited medium capacity, transmissions of nodes running different protocols can interfere. This chapter studies how to design node discovery algorithms for wireless networks which are robust against contending protocols on the shared medium. We pursue a conservative approach and consider a Byzantine adversary who disrupts the communication of our protocol in a worst-case fashion. This model also captures disruptions controlled by an adversarial *jammer*. We present optimal algorithms for several scenarios. The analytical findings are complemented by simulations providing evidence that the proposed protocols perform well in practice.

2.1 Device Discovery and Byzantine Disruptions

Wireless networks are ubiquitous and have become indispensable for many tasks of our daily lives. Due to the limited range of frequencies available for communication between wireless nodes such as laptops, PDAs or mobile phones, many wireless standards and protocols today operate on the same frequency bands, e.g., the ISM bands. One well-known and widely discussed example is WLAN and Bluetooth (i.e., *IEEE 802.15.2*), but there are many others. Such contending access of different protocols to the shared medium leads to collisions. While ultra wide band technology (UWB) may mitigate this problem and reduce interference, it is not always available or desirable.

This raises the question of how to devise protocols which are robust against transmissions of other protocols *by design*. In this chapter, we seek to shed light onto this question. We adopt a conservative approach and assume that a *Byzantine adversary* can disturb our algorithms in an arbitrary manner. This model comprises scenarios where an adversarial *jammer* attempts to slow down communication or even to stop it completely. Such jamming

attacks are a particularly cumbersome problem today: typically, a jamming attack does not require any special hardware and is hence simple and cheap.

This chapter focuses on networks without a fixed infrastructure, such as MANETs or sensor networks, which are organized in an ad hoc manner. A fundamental operation in dynamic ad hoc networks is the search of potential communication partners. In some sense, this operation is more difficult than other communication tasks, as the nodes do not have any information about each other *a priori*. Besides the lack of information on either hardware or medium access addresses of other nodes, concurrent transmissions—either of other nodes running the same protocol or other radio transmissions—lead to collisions and interference. In addition, by injecting a high level of noise, a jammer can slow down wireless communication significantly. Once two nodes have met, they may agree on certain communication or channel-hopping patterns (e.g., based on their medium access addresses) facilitating efficient interactions in the future. Thus, it is of utmost importance to solve this task as fast as possible.

A well-known existing protocol dealing with this problem is Bluetooth. It specifies an asymmetric way to connect and exchange information between devices such as mobile phones, cameras, GPS receivers or video game consoles. As a consequence, Bluetooth can be used to synchronize two devices as soon as they are within each other's transmission range, or to display the availability of a printer. Clearly, the device discovery time is highly relevant in these situations.

We study the problem of discovering communication partners in *multi-channel* networks, despite the presence of a Byzantine adversary. Concretely, we assume that the adversary corrupts t out of m available channels. We say that two nodes have successfully discovered each other if and only if two nodes are on the same channel, one transmitting, one receiving, there is no other node transmitting on this channel, and the channel is not jammed. In reality, nodes typically do not know *whether*, and *how many*, channels are corrupted. The goal is to devise algorithms solving the discovery problem efficiently *without knowledge of* t. We require that nodes are discovered very fast if t is small, and that the performance of the discovery algorithm *degrades gracefully* with increasing t. In other words, we want algorithms (oblivious to t) being *competitive* to a discovery algorithm knowing t.

Our main contribution are fast discovery algorithms performing well without knowledge of t and despite Byzantine disruptions. In particular, we describe a randomized algorithm which, in expectation, is at most a factor of $O(\log^2 m)$ slower than the best algorithm *knowing* t, for any t. We prove this to be optimal in the sense that this is the best ratio an algorithms that can be described by a probability distribution over the available channels can achieve. In addition, we study a scenario where the jammer chooses t according to a *probability density function* (PDF) which is known to the de-

vice discovery algorithm. Furthermore, we explain how to extend our results to a multiplayer setting and discuss alternative jammer models. In order to complement our formal analysis, we investigate the performance of our algorithms by in silico experiments.

Related Work

With the increasing popularity of wireless networks such as WLANs or sensor networks, security aspects and quality of service become more relevant. An important reason for disruptions are transmissions of other devices using different protocols. One widely studied example is WLAN and Bluetooth. In this case, several possible solutions have been discussed by the IEEE task force 802.15.2 [57], e.g., a non-cooperative coexistence mechanism based on adaptive frequency hopping.

The model we study is quite general and comprises many types of disruptions, such as interference [108] or jamming attacks. Resilience to jamming is crucial as jamming attacks can often be performed at low costs as there is no need for special hardware [11]. For these reasons, the jamming problem in wireless networks is intensively discussed both in practice and in theory ([30, 73, 89, 112, 113]). While some researchers focus on how such attacks can be performed [114], others concentrate on countermeasures [4]. In [77], it has been shown that using methods based on signal strength and carrier sensing, detecting sophisticated jammers is difficult. Moreover, a method based on packet delivery ratios cannot decide unambiguously whether link problems are due to mobility, congestion or jamming.

The threat of jamming attacks can be mitigated by appropriate physical layer technologies. E.g., spread spectrum techniques can be used, rendering it more difficult to detect the start of a packet fast enough in order to jam it. Unfortunately, one of the most widely deployed protocols, 802.11, has only small spreading factors [11]. In fact, it has recently been shown that the MAC protocol of 802.11 can be attacked by simple and oblivious jammers [15]. Many research projects deal with jammers on the MAC layer. For instance in [26], a coding scheme for fast frequency hopping is presented. If the adversary does not know the hopping sequence it can disturb only a subset of transmissions due to energy constraints. Alternative solutions include channel surfing and spatial retreat [114], or mechanisms to hide messages [112].

The jamming problem also raises interesting algorithmic questions. Gilbert et al. [47] investigate the *efficiency* of an adversary. The authors define the *jamming gain* of a given protocol P which compares the disruption duration of a computation to the adversary's cost. They find that even the uncertainty introduced by the possibility of adversarial broadcasts is sufficient to slow down many protocols. In [64] a model where the adversary has a *limited energy budget* is considered; the paper studies how to achieve

global broadcasts if the adversary is allowed to spoof addresses. In [94], fault-tolerant broadcasting under *probabilistic* failures is studied. Dolev et al. [34] analyze multi-channel networks, as we do. They present presents tight bounds for the running time of the ϵ-*gossip problem* on multi-channel networks. In [35], Dolev et al. describe a randomized protocol that allows nodes to exchange authenticated messages despite a malicious adversary that can cause collisions and spoof messages. Awerbuch et al. [11] present a MAC protocol for single-hop networks that is provably robust to adaptive adversarial jamming. The jammer can block a $(1-\epsilon)$-fraction of the time steps, but it has to make decisions before knowing the actions of the nodes for this step. Several algorithms are presented which, e.g., allow to elect a leader in an energy efficient manner.

In contrast to the work discussed above, we focus on the *bootstrap problem* where a node has to find other nodes in its range. This device discovery problem has been intensively studied in literature. In [19], randomized backoff protocols are proposed for a single broadcast channel. However, their algorithms are not directly applicable in wireless networks where unlike in traditional broadcast systems such as the Ethernet, collisions may not be detectable. In [72], probabilistic protocols for Bluetooth node discovery are investigated, where the nodes seek to establish one-to-one connections. In [6] and [7], protocols for single and multi channel ad hoc networks are described. However, none of these papers attend to (adversarial) disruptions.

2.2 Model and Definitions

Suppose we are given a shared medium consisting of m channels $c_1, ..., c_m$. There may be an adversary with access to the medium. We adopt a worst-case perspective assuming that an adversary always blocks those $t < m$ channels which maximize the discovery time of a given algorithm.

We aim at devising discovery protocols that are efficient despite these circumstances. Typically, the number of jammed channels t is not known to the discovery algorithm. Consequently, our main objective is to devise algorithms which are optimal with respect to *all* t. In other words, an algorithm *ALG* should solve the node discovery problem efficiently if t is small, and "*degrade gracefully*" for larger t.

For the analysis of the algorithms we investigate a *slotted* model where time is divided into synchronized time slots. However, note that all our results hold up to a factor of two in unslotted scenarios as well, due to the standard trick introduced in [99] for the study of slotted vs. unslotted ALOHA. In [99], it is shown that the realistic unslotted case and the idealized slotted case differ only by a factor of two. The basic intuition is that a single packet can only cause interference in two consecutive time-slots. Using the same argument, analyzing our algorithms in an "ideal" setting with synchronized

timeslots leads to results which are only a factor two better compared to the more realistic unslotted setting.

In each time slot, every node can choose one channel and decide whether it wants to listen on this channel or to transmit some information (e.g., its ID or a seed for its hopping pattern sequence) on the channel.

We assume that the nodes cannot detect collisions (no-CD). Moreover, they cannot differentiate between situations where (a) no node transmits on a channel, (b) several nodes transmit simultaneously on the same channel, or (c) the channel is jammed.

We say that two nodes v_1 and v_2 have *discovered* each other successfully if and only if the three following conditions are met:

1. v_1 and v_2 are on the same channel c
2. v_1 is in listening mode and v_2 transmits its contact information on c, or vice versa
3. channel c is not jammed

Since nodes cannot know, whether there are other nodes in their transmission area, we count the number of time slots until a successful discovery from the point in time when all of them are around (*discovery time*). We mainly constrain ourselves to the *two node case*.

The node diwithif we restrict ourselves to deterministic algorithms. In a scenario where all nodes are identical and do not have anything (e.g., IDs) to break the symmetry, two problems arise even in the absence of a jammer: (1) if two nodes follow a deterministic hopping pattern, they may never be on the same channel in the same slot; (2) even if the nodes meet, choosing deterministically whether to send or listen for announcements in this slot may always yield situations where both nodes send or both nodes listen. One way to break the symmetry is to allow nodes to generate random bits. Alternatively, one may assume that the two nodes which want to discover each other already share a certain number of bits which are unknown to the jammer. Due to these problems, we focus on *randomized* algorithms.

We assume that every node runs the same algorithm, only decisions based on random experiments differ. We investigate the class of randomized algorithms that can be described by a probability distribution over the channels, i.e., in each round, a channel is selected for communication according to a probability distribution. This has the advantage that the algorithms work well even if the nodes do not start the discovery process simultaneously.

We strive to find algorithms that perform well for every possible number of jammed channels. To this end, we define a measure that captures the loss of discovery speed due to the lack of knowledge of the number of channels the adversary decides to jam.

Definition 2.1 (Competitiveness)**.** *In a setting with t jammed channels, let T^t_{REF} be the expected discovery time until two nodes discover each other for an optimal randomized algorithm REF which has complete knowledge of t. Let T^t_{ALG} be the corresponding expected discovery time of a given algorithm ALG. We define the*

$$\text{competitive ratio} \quad \rho := \max_{0 \le t \le m-1} \frac{T^t_{ALG}}{T^t_{REF}}.$$

The smaller the achieved competitive ratio ρ, the more efficient the discovery algorithm.

2.3 Algorithms for Device Discovery

To initiate our analysis, we first consider device discovery algorithms for the case where the total number of jammed channels t is known. Subsequently, we present several algorithms for devices with less knowledge on the adversaries decisions.

Known t

In our model, a node has to select a channel c and decide whether to send or listen on c in each round. Let us determine the best strategy if t is known. As we will compare our algorithms which do not know t to this strategy, we will call this reference point algorithm REF. For two nodes which have never communicated before, it is best to send or listen with probability 0.5. The following lemma derives the optimal distribution over the channels.

Lemma 2.2. *Let m denote the total number of channels and assume t, the number of jammed channels, is known. If $t = 0$ the best strategy is to use one designated channel for discovery. If $0 < t \le m/2$ then the expected discovery time is minimized for an algorithm REF choosing one of the first $2t$ channels uniformly at random. In all other cases, the best strategy for REF is to chose each channel with probability $1/m$. Thus, REF has a expected discovery time of*

$$\begin{aligned} 2 \quad & \text{if } t = 0, \\ 8t \quad & \text{if } t \le m/2, \\ 2m^2/(m-t) \quad & \text{if } t > m/2. \end{aligned}$$

Proof. Let p_i denote REF's probability of choosing channel c_i. Without loss of generality, assume that the channels are ordered with decreasing probability of REF, i.e., $1 \ge p_1 \ge p_2 \ge ... \ge p_m \ge 0$. Let λ be the smallest i for which $p_i = 0$, in other words, REF uses λ channels for discovery. Clearly, if

$\lambda < t+1$, the expected discovery time is infinite, and hence, we can concentrate on algorithms for which $\lambda \geq t+1$.

According to our worst-case model, the jammer blocks the channels $c_1, \ldots, c_t$. It holds that $\sum_{i=t+1}^{\lambda} p_i \leq \frac{\lambda - t}{\lambda}$, where $(\lambda - t)/\lambda$ is equal to the sum of the channel probabilities of the channels $c_{t+1}, \ldots, c_\lambda$ when the probability distribution over the first λ channels is uniform. That is, by cutting some probability from those channels with probability greater than $1/\lambda$ and distribute it over the other channels, the total probability of success will increase. Therefore, the expected discovery time is minimized for uniform probability distributions. As soon as $\sum_{i=t+1}^{\lambda} p_i = \frac{\lambda - t}{\lambda}$, we cannot further redistribute the probabilities without decreasing the overall probability of success since the jammer always blocks the t most probable channels.

It remains to show that $\lambda = \min(2t, m)$ maximizes the probability of success. As the first t channels are jammed and the probability to be chosen is $p_i = 1/\lambda$ for each channel, the probability for a successful meeting is

$$\mathbb{P}[\text{success}|t] = \tfrac{1}{2} \sum_{i=t+1}^{\lambda} \tfrac{1}{\lambda^2} = \tfrac{\lambda - t}{2\lambda^2}.$$

This probability is maximized for $\lambda = 2t$. If fewer channels are available, i.e., $2t > m$, the best decision is to pick any of the m channels with probability $1/m$. Since the execution in one time slot is independent from the execution in all other time slots, the expected discovery time is then given by the inverse of the success probability. □

Two Simple Algorithms

The simplest randomized algorithm chooses one of the available m channels uniformly at random in each round. The expected expected discovery time of this algorithm UNI is $2m^2/(m-t)$. Hence the competitiveness of UNI is $\rho_{UNI} = m$, reached when $t = 0$. In other words, if there are no blocked channels, the performance of this algorithm is poor.

Since we aim at being competitive to REF for any number of jammed channels t, we examine more sophisticated algorithms. Observe that for small t, selecting a channel out of a small subset of channels is beneficial, since this increases the probability that another node is using the same channel. On the other hand, for large t, using few channels is harmful, as most of them are jammed. One intuitive way to tackle the device discovery problem is to use a small number of estimators for t. In each round, we choose one of the estimators according to a probability distribution and then apply the optimal algorithm for this "known" $\hat{t}$, namely algorithm REF. In the following, we will refer to the set of channels for such a $\hat{t}$, i.e., channels $c_1, ..., c_{2\hat{t}}$, as a *class* of channels. Note that any such algorithm has to include the class $\hat{t} = m/2$, otherwise the expected discovery time is infinity.

As an example, consider an algorithm ALG_3 guessing t to be either $\hat{t} = 1$, $\hat{t} = \frac{\sqrt{m}}{2}$ or $\hat{t} = \frac{m}{2}$ with equal probability. I.e., ALG_3 chooses a random

channel out of one of the *classes* $\{c_1, c_2\}$, $\{c_1, \ldots, c_{\sqrt{m}}\}$ or $\{c_1, \ldots, c_m\}$ in each round.

Theorem 2.3. *The competitive ratio of A_3 is*

$$\rho_3 = \frac{9m}{4(\sqrt{m}-1)} \in O(\sqrt{m}).$$

Proof. The exact probability for a successful meeting, given the number of channels the nodes take into account, is given in Figure 2.1. Note that we omit the factor 0.5 which is due to the constraint that the nodes should not be either both listening or both sending. Summarized, we have the following success probabilities.

$$\begin{aligned}
\frac{1}{18}\left(\frac{\sqrt{m}-t}{m} + 2\frac{\sqrt{m}-t}{m\sqrt{m}} + \frac{m-t}{m^2} + 0.5 + \frac{2}{\sqrt{m}} + \frac{2}{m}\right) &\quad \text{for} \quad t < 2 \\
\frac{1}{18}\left(\frac{\sqrt{m}-t}{m} + 2\frac{\sqrt{m}-t}{m\sqrt{m}} + \frac{m-t}{m^2}\right) &\quad \text{for} \quad 2 \le t < \sqrt{m} \\
\frac{1}{18}\left(\frac{m-t}{m^2}\right) &\quad \text{for} \quad \sqrt{m} \le t
\end{aligned}$$

Since the outcome of each round is independent of all previous rounds, the expected discovery time is the inverse of the success probability. It remains to compute the competitiveness ρ_3, i.e., the maximum ratio of the expected discovery time of ALG_3 and REF. It is easy to verify that the maximum ratio is reached if $t = \sqrt{m}$. Thus ρ_3 is $9m/4(\sqrt{m}-1)$, completing the the proof. □

		d_1 = 2		d_1 = $\sqrt{m}$		d_1 = m	
d_2 = 2		$\frac{1}{2}$	if $t < 2$	$\frac{1}{\sqrt{m}}$	if $t < 2$	$\frac{1}{m}$	if $t < 2$
		0	if $t \ge 2$	0	if $t \ge 2$	0	if $t \ge 2$
d_2 = $\sqrt{m}$		$\frac{1}{\sqrt{m}}$	if $t < 2$	$\frac{\sqrt{m}-t}{m}$	if $t < \sqrt{m}$	$\frac{\sqrt{m}-t}{m\sqrt{m}}$	if $t < \sqrt{m}$
		0	if $t \ge 2$	0	if $t \ge \sqrt{m}$	0	if $t \ge \sqrt{m}$
d_2 = m		$\frac{1}{m}$	if $t < 2$	$\frac{\sqrt{m}-t}{m\sqrt{m}}$	if $t < \sqrt{m}$	$\frac{m-t}{m^2}$	
		0	if $t \ge 2$	0	if $t \ge \sqrt{m}$		

Figure 2.1: Meeting probability when applying ALG_3 (Factors 0.5 omitted).

This shows us that ALG_3 is a factor of $\Theta(\sqrt{m})$ better than UNI with respect to its competitiveness.

Class Algorithms

The previous paragraphhas motivated the study of class algorithms using several estimators for t. We pursue this idea further and investigate the optimal number of classes for the family of algorithms selecting the estimator for the next round uniformly at random among k guess classes $\widehat{t_1} \leq ... \leq \widehat{t_i} \leq ... \leq \widehat{t_k}$ for t for $k \leq m/2$. The algorithm chooses each such class i with a uniform probability, and subsequently selects a channel to transmit uniformly at random from a given set of $2\widehat{t_i}$ channels. We concentrate on algorithms ALG_k where the guesses grow by constant factors, i.e., whose estimations for $\widehat{t}$ are of the following magnitudes: $\widehat{t} = m^{1/k}, ..., m^{i/k}, ..., m$. We begin by deriving a bound on the expected discovery time of ALG_k.

Theorem 2.4. *Let m denote the number of channels and let $t < m$ be the number of jammed channels. Let $\beta_1 = \left\lfloor \frac{k \cdot \ln(t)}{\ln m} \right\rfloor$, $\beta_2 = m^{-\frac{\beta_1}{k}}$ and $\beta_3 = 2\beta_1 - 2k - 1$ for some integer value $k \leq m/2$. The expected discovery time of ALG_k is*

$$\frac{2k^2 m(m^{1/k} - 1)^2}{m^{\frac{1}{k}}\beta_3 - \frac{t}{m} + \beta_3 + \beta_2\left(m^{\frac{k+1}{k}} + 2t + m - t\beta_2 m\right)}.$$

Proof. Consider a time slot where node v_1 chooses class i_1 and assume node v_2 chooses class i_2 in the same round. If a discovery is possible in this time slot, we have $i_1 \geq i_2 > \frac{k \cdot \ln(t)}{\ln m}$. The second inequality is due to our requirement that $m^{i_2/k} > t$; otherwise the devices cannot find each other since the estimator $\hat{t}$ of at least one device smaller than $t/2$. The probability that the two nodes successfully meet in this round is

$$p(i_1, i_2, t) = \frac{m^{i_2/k} - t}{m^{i_1/k} m^{i_2/k}} = \frac{1}{m^{i_1/k}} - \frac{t}{m^{(i_1+i_2)/k}}.$$

The overall success probability in some round is given by

$$\mathbb{P}[\text{success}|t] = \frac{1}{2k^2}\left(\sum_{i_1=\beta_1+1}^{k}\left(\sum_{i_2=\beta_1+1}^{i_1-1} p(i_1, i_2, t) + \sum_{i_2=i_1}^{k} p(i_2, i_1, t)\right)\right).$$

Expanding the sums leads to $\mathbb{P}[success|t] := (m^{\frac{1}{k}}(2\beta_1 - 2k - 1) - \frac{t}{m} + 2k - 2\beta_1 - 1 + m^{-\frac{\beta_1}{k}}(m^{\frac{k+1}{k}} + 2t + m - tm^{1-\frac{\beta_1}{k}}))/(2k^2 m(m^{1/k} - 1)^2)$, of which the expected discovery time can be derived. □

Since we are particularly interested in an algorithm's competitiveness, we can examine the ratio achieved by this algorithm for $k = \lfloor \log m \rfloor$. We give a brief sketch of the derivation.

We distinguish three cases for t. If $t = 0$, the ratio is $\Theta(\log m)$, verifiable by calculating $2/\mathbf{P}[success|t = 0]$. For $t \in [1, \ldots, m/2]$, the expected running time is

$$O(\log^2 m \cdot mt/(m - \log m)),$$

hence the ratio is $O(\log^2 m)$. It remains to consider $t > m/2$. The expected discovery time of $ALG_{\log m}$ is

$$\frac{2m^2 \log^2 m}{m - t},$$

compared to REF needing $2m^2/(m - t)$ time slots in expectation. Thus the ratio is at most $O(\log^2 m)$ for $ALG_{\log m}$. Interestingly, as we will see later, this is asymptotically optimal even for general randomized algorithms.

Corollary 2.5. *$ALG_{\log m}$ has a competitive ratio of at most $O(\log^2 m)$.*

Optimal Competitiveness

We have studied how to combine algorithms tailored for a certain estimated t in order to construct efficient node discovery protocols. In particular, we have derived the execution time for a general class of algorithms ALG_k. This raises two questions: What is the best competitive ratio achieved by ALG_k with the best choice of k? How much do we lose compared to *any* algorithm solving the device discovery problem by focusing on such class estimation algorithms ALG_k only?

In the following, we adopt a more direct approach, and construct an optimal algorithm using a probability distribution $\overrightarrow{p} = (p_1, \ldots, p_m)$, i.e., choosing a channel i with probability p_i, where $p_1 \geq p_2 \geq \ldots \geq p_m \geq 0$. In other words we have to find $\overrightarrow{p}$ yielding the lowest possible competitiveness. From this analysis, we can conclude that no loss incurs when using on class algorithms ALG_k , i.e., there is a class algorithm, $ALG_{\log n}$ with an asymptotically optimal competitive ratio.

Recall that the best possible expected discovery time (cf. Lemma 2.2) if t channels are jammed and if t is known. Thus, in order to devise an optimal algorithm OPT, we need to solve the following optimization problem.

$$\min \rho = \min_{\overrightarrow{p}} \max_{0 \leq t < m} \frac{T^t_{ALG}}{T^t_{REF}},$$

$$\text{where } \frac{T^t_{ALG}}{T^t_{REF}} = \begin{cases} \frac{1}{\sum_{i=1}^m p_i^2} & \text{if } t = 0 \\ \frac{1}{4t \sum_{i=t+1}^m p_i^2} & \text{if } t \leq m/2 \\ \frac{m-t}{m^2 \sum_{i=t+1}^m p_i^2} & \text{if } t > m/2. \end{cases}$$

In addition, it must hold that $\sum_{i=1}^m p_i = 1$, and $p_1 \geq p_2 \geq ... \geq p_m \geq 0$.

We simplify the $\min\max\rho$ objective function to $\min\rho$ by generating the following optimization system.

$$\begin{aligned}
&\min\rho \ \text{ such that}\\
&t=0: && \tfrac{1}{\sum_{i=1}^{m} p_i^2} \le \rho && (1)\\
&1\le t\le m/2: && \tfrac{1}{4t\sum_{i=t+1}^{m} p_i^2} \le \rho && (2)\\
&t> m/2: && \tfrac{m-t}{m^2\sum_{i=t+1}^{m} p_i^2} \le \rho && (3)\\
&\text{and} \quad \sum_{i=1}^{m} p_i = 1, \quad p_1\ge p_2\ge ...\ge p_m\ge 0.
\end{aligned}$$

Observe that ρ is minimal if equality holds for all inequations in (1), (2) and (3). This yields an equation system allowing us to compute the values p_i. Thus the optimal channel selection probabilities are

$$\begin{aligned}
p_1 &= \sqrt{\tfrac{7}{8\rho}}\\
p_i &= \sqrt{\tfrac{1}{4i(i-1)\rho}} \quad \text{for i}\in[2,m/2],\\
p_j &= \tfrac{1}{m}\cdot\sqrt{\tfrac{1}{\rho}} \quad \text{for } j> m/2.
\end{aligned}$$

Due to the constraint $\sum_{i=1}^{m} p_i = 1$, the competitiveness is

$$\rho = \tfrac{1}{4}\left(1+\sqrt{7/2}+\textstyle\sum_{i=2}^{m/2} 1/\sqrt{i(i-1)}\right)^2.$$

Since

$$H_{m/2-1} = \sum_{i=2}^{m/2} 1/\sqrt{(i-1)^2} > \sum_{i=2}^{m/2} 1/\sqrt{i^2} = H_{m/2} - 1,$$

where H_i is the i^{th} harmonic number, it holds that $\rho\in\Theta(\log^2 m)$. Thus, we have derived the following result.

Theorem 2.6. *Algorithm OPT solves the device discovery problem with optimal competitiveness*

$$\tfrac{1}{4}\left(1+\sqrt{7/2}+\textstyle\sum_{i=2}^{m/2} 1/\sqrt{i(i-1)}\right)^2 \in \Theta(\log^2 m).$$

As mentioned above, the class algorithm $ALG_{\log m}$ features an asymptotically optimal competitiveness of $\Theta(\log^2 m)$ as well.

Optimality for Known Probability Distribution of t

In the previous section, we have described an algorithm which solves the discovery problem optimally for unknown t. In the following, we continue our investigations in a slightly different model where the algorithm has a rough estimation on the total number of jammed channels. Concretely, we assume that an algorithm has an a priori knowledge on the probability distribution of the total number of jammed channels: Let $p(0), p(1), \ldots, p(i), \ldots, p(m)$ be the probability that i channels are jammed. We know from Section 2.3 that if $t = i \leq m/2$ is known, the optimal discovery time is $8i$ in expectation. We want to devise an algorithm ALG_{PDF} which estimates t using the distribution $x_0, x_1, ..., x_{m/2}$ over the classes estimating $\widehat{t} = i$, and minimizing the expected total execution time.

Let p_i denote the success probability for $t = i \leq m/2$, i.e., i channels are jammed. For the two classes j and l used by the two nodes, we have a success probability of

$$\frac{\max\{\min\{2j - i, 2l - i\}, 0\}}{(2 \cdot 2j \cdot 2l},$$

since the nodes can only meet on unjammed channels. In order to compute p_i, we need to sum over all possible pairs of classes multiplied with the probability of selecting them.

$$\begin{aligned} p_i &= \mathbb{P}\left[\text{success} | t = i\right] \\ &= \sum_{j>i/2}^{m/2-1} \sum_{l>i/2}^{m/2} x_j x_l \frac{\min(2j - i, 2l - i)}{8jl} \\ &= \sum_{j>i/2}^{m/2-1} \sum_{l=j+1}^{m/2} 2x_j x_l \frac{2j - i}{8jl} + \sum_{j=i}^{m/2} x_j^2 \cdot \frac{2j - i}{8j^2}. \end{aligned}$$

For $t > m/2$, the expected discovery time is $\frac{m^2}{(m-t)x_{m/2}^2}$. This leaves us with the following optimization problem:

$$\begin{gathered} \min \left[\sum_{i=0}^{m/2} p(i)/p_i + \sum_{i=m/2+1}^{m} p(i) \cdot \frac{m^2}{x_{m/2}^2 (m-i)} \right] \\ \text{subject to } \sum_{i=0}^{m/2} x_i = 1. \end{gathered}$$

Unfortunately, this formulation is still non-linear. However, there are tools available that can compute the optimal x_i's of ALG_{PDF} numerically using this formulation.

2.4 Multi-Player Setting

Scenarios with more than two nodes raise many interesting questions. One could try to minimize the time until the first two nodes have met, the time until a given node has found another given node, or the time until all nodes have had at least one direct encounter with all other nodes in the vicinity. In practice, instead of computing a complete graph where each pair of nodes has interacted directly, it might be more important to simply guarantee connectivity, i.e. ensure the existence of acquaintance paths between all pairs of nodes. In some of these models, it is beneficial to coordinate the nodes and divide the work when they meet.

We leave the study of node coordination strategies for future research. However, in the following, we want to initiate the multi-player analysis with a scenario where the total number of nodes n and the total number of jammed channels t is known, and where a node u wants to find a specific other node v while other nodes are performing similar searches concurrently. Again, we assume a symmetric situation where all nodes execute the same randomized algorithm.

Theorem 2.7. *Let n be the number of nodes and assume t, the number of jammed channels, is known. If there are $\Omega(n+t)$ channels available, the asymptotically best expected discovery time is $\Theta(n+t)$. The algorithm selecting one of the first $max(2t, 2n)$ channels uniformly at random and sending with probability $1/2$ achieves this bound.*

Proof. Let the i^{th} channel be selected with probability p_i, and assume a given node sends (or listens) on the channel with probability p_s (or with probability $1-p_s$). By the same argument as presented in the previous section, there exists a randomized algorithm minimizing the expected node discovery time by selecting $p_i = 1/k \ \forall i < k$ for some variable k. The discovery probability if k channels are used is given by

$$(k-t)k^{-2}2p_s(1-p_s)(1-p_s/k)^{n-2}.$$

Thus, it remains to compute p_s and k. Let us start with the last factor of the success probability. Using the fact that $(1-x/n)^n > e^{-x}$, we can guarantee that the term $(1-p_s/k)^{n-2}$ is asymptotically constant, if $p_s/k \propto n$ (condition (1)). Clearly, we have to choose $k > t$ to ensure that a meeting can happen (condition (2)). Asymptotically, the expected discovery time is in $\Theta(t+n)$, regardless of the precise choice of k and p_s—as long as the conditions (1) and (2) are satisfied. Concretely, setting $p_s = 1/2$ and $k = \max(2t, 2(n-2))$ leads to an asymptotically optimal expected discovery time. □

In reality, nodes typically do not know the number of nodes that are active in the same area simultaneously. What happens if we apply the optimal

strategy for two nodes devised in Section 2.3, even though there might be several other nodes? Using the same arguments as in Section 2.3, we can derive that every node executing algorithm OPT is asymptotically optimal as well.

Corollary 2.8. *Let n be the number of nodes and t the number of jammed channels. Assume that n and t are unknown to the nodes. Algorithm OPT from Section 2.3 achieves an asymptotically optimal competitiveness.*

2.5 Alternative Jammer Models

So far, we have focused on a worst case jammer model where the jammer blocks those t channels used with the highest probability by the nodes. However, depending on the hardware of the jammer, many alternative models can be of interest too. We can categorize jammers into *pure jammers* which do not listen and adapt to the traffic generated by the nodes, and *listening jammers* which listen on some channels to check whether a node is currently transmitting there. See Figure 2.2 for an overview.

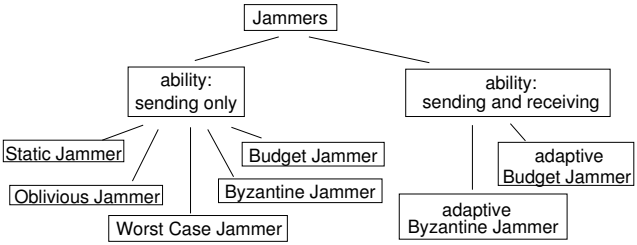


Figure 2.2: Jammer models.

We can distinguish between different types of pure jammers. In the simplest case, the jammer jams a fixed set of t channels (*static jammer*). If the set of jammed channels is known to the nodes, discovery is of course simple. Otherwise our optimal algorithm solves the problem efficiently. Alternatively, an *oblivious jammer* may jam t channels selected at random in every round. If these channels are chosen uniformly at random, then a good strategy to discover a node is to simply wait on some fixed channel until it is free. However, an oblivious jammer is more tedious than a static jammer during the subsequent communication: while in the case of static jammers once a free channel has been found nodes can communicate on this channel without interruptions, an oblivious jammer continues to have a non-zero probability to interfere. A slightly more complex jammer is a *Byzantine jammer* which might either remain on the same channels or choose alternative channels to

jam at random. Since we consider pure jammers only, this decision cannot depend on the actions of the nodes. A Byzantine jammer is interesting, as applying the algorithms devised for static or oblivious jammers might turn out not to be a successful strategy. Finally, another pure jammer model is the *budget jammer*. The budget jammer amortizes its cost over a number of rounds. For example, it is allowed to jam a total of $t \cdot r$ channels over a period of r rounds. I.e., a jammer may jam much more than t channels in one round but then only a few in the next rounds. It can easily be seen that our optimal algorithm proposed in Section 2.3 has the same competitiveness when attacked by this adversary or a static jammer.

Listening jammers have the ability to check whether a node is currently transmitting on a given channel. We assume that in each round, the jammer can listen on t_s channels and jam on $t_j = t - t_s$ channels, where t_s (and hence t_j) can change dynamically over time. Of course we can examine Byzantine and budget jammers again. A number of new attacks are now possible, since the adversary can *adapt its strategy* to the behavior of the nodes, especially in the multi-player case. For instance, a jammer may reverse engineer the hopping pattern used by the nodes after they have met. An adaptive budget jammer on the other hand may use all its power to target and subsequently block a particular node during the communication phase.

We do not indulge into a more detailed discussion here. However, in summary, we want to emphasize that there exist many additional types of jammer models which may be of concern in practice. We believe that due to the rather pessimistic model studied in Section 2.3, our discovery algorithms perform well and the upper bounds on the expected discovery time hold for many alternative models.

2.6 Simulations

In order to complement our formal results, we conducted several *in silico* experiments to study the behavior of our algorithms in different settings. In this section, we discuss our main simulation results. If not mentioned otherwise, we examine a system with 128 channels (Bluetooth uses 79 channels, 32 for discovery) and we discuss the average discovery time of 1000 experiments.

Device Discovery

In a first set of experiments, we studied the average discovery time of the optimal algorithm OPT and the algorithm using a logarithmic number of estimators or classes ($ALG_{\log m}$), see also Section 2.3. A simple solution to the device discovery problem typically used in practice is to select the available channels uniformly at random. Therefore, we include in our plots the algorithm UNI which has a balanced distribution over the channels.

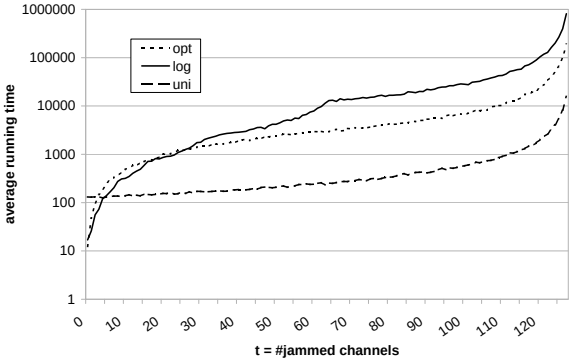


Figure 2.3: Average discovery time of OPT, $ALG_{\log m}$, and UNI as a function of the total number of jammed channels t.

Figure 2.3 shows that in case only a small number of channels is jammed, OPT and $ALG_{\log m}$ yield much shorter discovery time (around a factor ten for $t = 0$). However, as expected, the uniform algorithm UNI is much faster if a large fraction of channels are jammed.

The study of the algorithms' competitive ratio is more interesting. Figure 2.4 plots the ratios of the different algorithms' discovery time divided

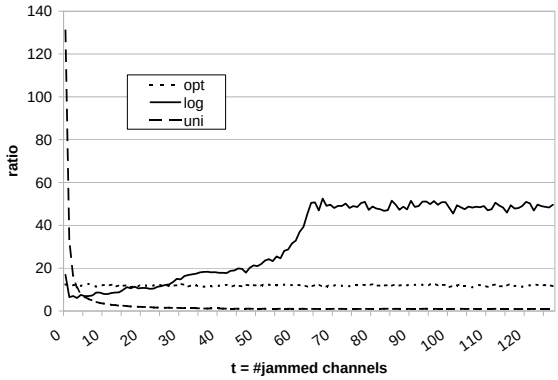


Figure 2.4: Competitive ratios of OPT, $ALG_{\log m}$, and UNI as a function of the total number of jammed channels t.

by the optimal running time if t is known achieved by REF (cf. Section 2.3). The figure shows that our optimal algorithm OPT has indeed a perfectly balanced competitiveness of around 12, independently of the number of jammed channels t. The uniform algorithm UNI is particularly inefficient for small t, but improves quickly for increasing t. However, over all possible values for t, UNI's ratio is much worse than that of OPT and $ALG_{\log m}$. Note that $ALG_{\log m}$ is never more than a constant factor off from the optimal algorithm OPT (a factor of around four in this example). The competitive ratio of $ALG_{\log m}$ reaches its maximum at $t > m/2$.

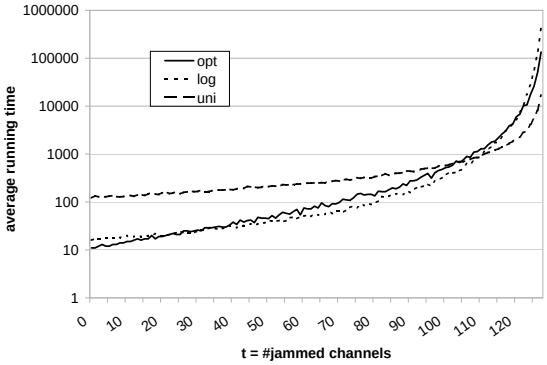


Figure 2.5: Average discovery time of OPT, $ALG_{\log m}$, and UNI as a function of the total number of jammed channels t. For this plot, the jammed channels are chosen *uniformly at random*.

So far, we have assumed a rather pessimistic point of view in our analysis and we considered a worst case adversarial jammer only. Figure 2.5 studies the algorithms in a setting where a random set of t channels is jammed. Clearly, OPT and $ALG_{\log m}$ perform much better than UNI even for quite a large number of jammed channels. Only if the number of jammed channels exceeds 100, the average discovery time is worse.

Microwave Case Study

Besides adversarial jamming attacks, a reason for collisions during the discovery phase is interference from other radio sources. It is well-known that microwave ovens interfere with Bluetooth channels (e.g., [68]), especially Bluetooth Channels 60-70. These channels are among the 32 channels that the Bluetooth protocol uses for discovery (called *inquiry* in Bluetooth speak). In other words, the Bluetooth protocol does not exploit the full range of

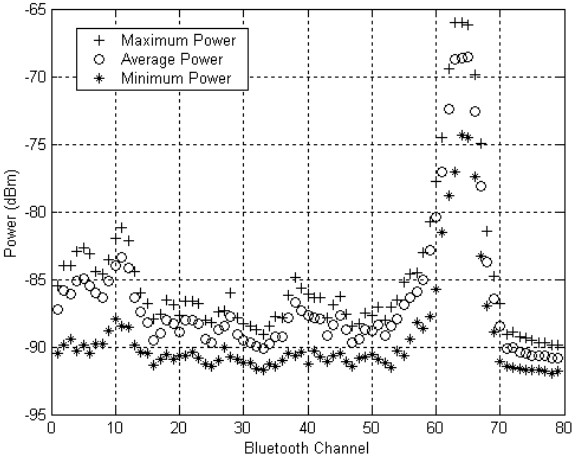


Figure 2.6: Interference experienced by each individual Bluetooth Channel when the receiver is 20 feet away from a Microwave Oven. This figure from [68] is used with the permission of the authors and is subject to IEEE copyright restrictions

available channels for discovery. The Bluetooth protocol is asymmetric, i.e., nodes either scan the inquiry channels from time to time, or they try to find nodes nearby.

We have conducted a case study modelling the presence of other nodes and a microwave oven. To this end, we simplified the Bluetooth inquiry protocol to its core device discovery algorithm. One node scans the channels constantly and the other node performs the Bluetooth inquiry frequency hopping pattern until they meet. Since Bluetooth only uses 32 out of the 79 available channels for discovery, our optimal algorithm is clearly in advantage by exploiting the whole range of frequencies. We ignore this advantage and consider the following set up: two nodes applying the Bluetooth inquiry protocol and two nodes executing the optimal algorithm for 32 channels seek to meet the node following the same protocol. We have counted the number of time slots Bluetooth and our optimal algorithm need until this meeting happens with and without interference by a microwave oven. We obtained the following results:

Microwave	BT	OPT
off	34.49	15.16
on	45.76	15.70

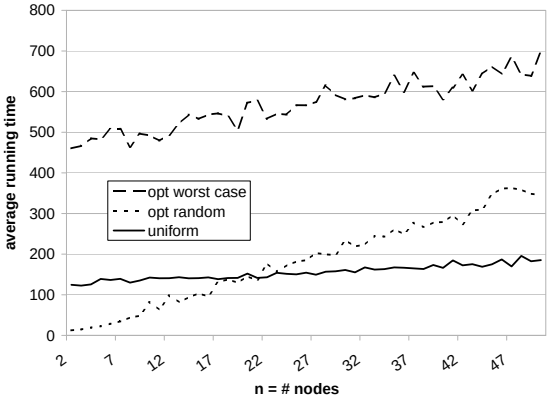


Figure 2.7: Comparison of the discovery time of OPT (once with randomly and once with worst-case jammed channels) to UNI if $t = 10$ channels are jammed and for different number of nodes executing the same algorithm concurrently.

There is a substantial difference between the performance of the two protocols, especially when considering that the Bluetooth protocol is asymmetric. Hence no collisions occur on the same channels in our setting with two Bluetooth nodes. In other words, our setting is punishing the optimal algorithm for being symmetric. We believe that there are many interesting scenarios where symmetry is required and protocols following a Bluetooth approach are not suitable.

Multi-Player Settings

The algorithms described in Section 2.3 are tailored to settings where two nodes want to meet efficiently despite a adversarial jammer. However, our analysis and our experiments show, that the number of time slots until two designated nodes meet increases linearly in the number of nodes in the vicinity. In large networks or times of high contentions the UNI algorithm performs much better. Thus, in these scenarios, it is beneficial to use this algorithm.

2.7 Concluding Remarks

The fast and robust discovery of other devices is one of the most fundamental problems in wireless computing. Consequently, a prerequisite to efficient

networking are algorithms with the twofold objective of allowing devices to find each other quickly in the absence of any interference, degrading gracefully under increasing disturbance. In other words, discovery algorithms that work well in different settings and under various conditions are necessary. This chapter has presented optimal algorithms for a very general, Byzantine model of communication disruptions. This implies that our algorithms can be used in many other scenarios with stronger assumptions on the nature of such disruptions. In other words, our algorithms can cope with incidental as well as with malicious interference. Furthermore, our algorithms are ideal candidates for energy and memory constrained sensor nodes as they are simple and fully distributed.

Other approaches, e.g., based on exponential search techniques can outperform our protocols if the adversary is static, i.e. does not change the number of blocked channels. Such protocols are not able to deal with situations where the number of blocked channels varies. Another disadvantage of the exponential search technique is the fact that, in contrast to our algorithm, it requires the nodes to start the discovery protocol at the same time.

Our results open many directions for future research. It is important to reason about how the first successful contact between two nodes can be used for more efficient future information exchange (e.g., by establishing a shared secret key), and how, subsequently, more complex tasks can be performed over the multi-channel system.

3

Uniform Power Scheduling

How long does it take to find an optimal schedule for a given set of communication links in a wireless ad-hoc network? Is this problem difficult – even in a simplified model? What if we do not need to schedule all communication links, but simply want to choose the most "valuable" ones? And how hard is it to produce a result which is not necessarily optimal, but only falls short of an optimal solution by a guaranteed factor? In this chapter, we study the computational complexity of constructing optimal schedules. In particular, we present NP-hardness results and approximation algorithms for two problems: Scheduling and One-Shot Scheduling. The first problem consists in finding a minimum-length schedule for a given set of links. The second problem receives a weighted set of links as input and consists in finding a maximum-weight subset of links to be scheduled simultaneously in one shot.

3.1 Geometry Matters

When studying wireless networks, the choice of the interference model is of fundamental significance. Not only has the selected model to incorporate the nature of real networks, but also to facilitate the development of rigorous reasoning.

One model of choice is the *abstract* Signal-to-Interference-plus-Noise-Ratio (or short, $SINR_A$) model. In the $SINR_A$ model, a signal is received successfully depending on the ratio of the received signal strength and the sum of the interference caused by nodes sending simultaneously (plus noise).

The wireless networking community usually adheres to the physical $SINR$ model ($SINR_P$). In this inherently geometric model, the nodes live in space, and the gain (or signal attenuation) between two nodes is determined by the distance between the two nodes. In particular, a signal fades with the distance to the power of alpha, alpha being the so-called path-loss parameter.

$SINR_P$ makes some simplifying assumptions, such as perfectly isotropic radios, no obstructions, or a constant ambient noise level. The more general $SINR_A$ model allows arbitrary values in the gain matrix among the participating nodes of a wireless network. This model can capture some of the properties $SINR_P$ is missing. On the other hand, $SINR_A$ is not all that realistic either, since in reality, if a node u is close to a node v, which in turn is close to a node w, then u and w will also be close. So the entries in the gain matrix will be constrained by the other entries. Thus, t$SINR_P$ is too optimistic, whereas $SINR_A$ is too pessimistic. Hence, a real network is positioned somewhere between the $SINR_P$ and the $SINR_A$ model.

When studying algorithms or protocols, upper bounds should be derived for the pessimistic model, as an algorithm for a strictly[1] more pessimistic model will also work for reality. However, also the converse is true: If one is interested in lower bounds (impossibility results or capacity constraints), one must use the optimistic model. A strictly more optimistic model guarantees that results are applicable in practice.

In this chapter we study two optimization problems in wireless networks: Scheduling and One-Shot Scheduling. Apart from presenting approximation algorithms, our main result is the proof of hardness of these problems. In particular, we formally prove that Scheduling and One-Shot Scheduling are both NP-hard in the $SINR_P$ model. Since the $SINR_P$ model is weaker than reality, this implies that one cannot compute an optimal schedule of wireless requests in practice, unless $P = NP$.

To the best of our knowledge, these are the first NP-hardness proofs for the $SINR_P$ model.[2] As we will discuss in the next section in more detail, there have been various NP-completeness proofs for wireless networks model, in particular for so-called unit disk graphs (UDG) or for the $SINR_A$ model. In contrast to our work, these proofs are graph-based. In an orthodox $SINR_A$ proof one establishes an arbitrary gain matrix between the participating nodes of a wireless network, giving $O(n^2)$ degrees of freedom. In particular, this allows to build a graph, as the gain between any two nodes can be set to either 1 ("link") or 0 ("no link"). One ends up with a standard graph, and it trivially follows that e.g. scheduling is as hard as coloring in graphs. A similar argument holds for proofs for the UDG model.[3]

In reality, however, gain cannot be chosen arbitrarily. As we argued

[1] Note that models are rarely strictly harder than reality; $SINR_A$ is a typical example, as $SINR_A$ does not include several difficulties of reality, e.g. short-term fading.

[2] In order to prove that the problems at hand are *NP-complete* as well, we have to prove that they in the complexity class NP. For some operations on integers it is not yet clear whether they can be computed efficiently by a Turing machine. E.g., it is not known how a sum of of square roots of integers can be compared quickly to an integer [90]. Since our model requires the computation of roots of integers, we do not know whether scheduling and related problems are in NP. If we assume the Real RAM model (often used in computational geometry), all our computations can be implemented efficiently.

[3] Not surprisingly as the G in UDG stands for graph.

before, the triangular inequality makes all the entries in the gain matrix interdependent. If we turn to the $SINR_P$ model, we must choose positions of the nodes in space (e.g. in a plane), which determines the attenuation between two nodes, giving only $O(n)$ degrees of freedom. Arguing that two nodes cannot transmit concurrently in a schedule becomes much harder, since the nodes all influence each other. This is what intuitively makes the problem harder. In the $SINR_P$ model, one must always deal with the complete (weighted) graph; this asks for a different kind of proof.

The problem of scheduling link transmissions in a wireless network in order to optimize one or more of performance objectives (e.g. throughput, delay, fairness or energy) has been a subject of much interest over the past decades.

Previous Complexity Results

An issue of prime importance is the complexity of scheduling problems. As has already been argued in the introduction, there have been various NP-completeness proofs for wireless networks. To the best of our knowledge, these proofs are either built for the UDG model [56, 69], or for the abstract $SINR$ model ($SINR_A$), and present reductions without a geometric representation. A typical such proof establishes an arbitrary gain matrix between the participating nodes, which results in a standard graph. Afterwards, the hardness is proved by a reduction from graph coloring, for example [20].

The joint problem of power control and scheduling with the objective of minimizing the total transmit power subject to the end-to-end bandwidth guarantees and the bit error rate constraints of each communication session is addressed by Kozat et al. in [66]. They prove their problem to be NP-complete by using a reduction from integer programming under the assumption that the values of the gain matrix can be chosen arbitrarily.

Similarly, Leung and Wang [75] prove that the problem of maximizing data throughput by adaptive modulation and power control while meeting packet error requirements is NP-complete under the assumption that the values of the gain matrix are arbitrary.

Another problem is proposed by Chatterjee et al. in [79] as the power constrained discrete rate allocation problem. A solution finds the rates at which the base station must transmit to each user including $SINR$ constraints. They prove that this problem is NP-complete for CDMA data networks by a reduction from the Knapsack problem using a gain matrix with gain value 1 for all links.

The problem of scheduling broadcast requests has been studied by Ephremides and Truong [39]. They show that in a generalized, *non-geometric* model, finding an optimal schedule is NP-complete, if no interference is tolerated. Other aspects of scheduling and power control using an arbitrary gain matrix are studied for instance in [20, 21, 32, 93, 96, 97].

One of the very few lower bounds for the $SINR_P$ model is due to Gupta and Kumar [53]. They analyze the overall capacity of ad-hoc networks in the $SINR_P$ model from an *information theoretic* perspective, and prove that a wireless network comprised of n nodes cannot provide a throughput of more than $\Theta(1/\sqrt{n})$.

Graph-Based Scheduling Algorithms

Of course the design of efficient algorithms for scheduling has been explored as well. In order to compute a time-schedule such that spatial reuse is maximized, most of the proposed schemes are based on traditional graph-theoretic models. They use a graph representation of a wireless network, modeling interference by some (often binary) graph property. For example, a set of "interference-edges" might be defined, containing pairs of nodes within a certain distance to each other, thus modeling interference as a local measure.

Graph-based scheduling algorithms usually employ an implicit or explicit coloring strategy, which neglects the aggregated interference of nodes located farther away. A variety of centralized and decentralized approximation algorithms have been proposed and their quality analyzed for this kind of model [54, 71, 84, 98, 102]. Most recently, Brar et al. [23] present a scheduling method that is based on a greedy assignment of weighted colors. Although these algorithms present extensive theoretical analysis, they are constrained to the limitations of a model that does not reflect the real nature of wireless networks. In particular, such graph-based models ignore the accumulated interference of a large number of distant nodes.

In [17, 50, 51], it is argued that the performance of graph-based algorithms is inferior to algorithms in more realistic $SINR$ models. More recently, Moscibroda et al. [86] show experimentally that the theoretical limits of any protocol, which obeys the laws of graph-based models, can be broken by a protocol explicitly defined for the $SINR_P$ model.

Scheduling Algorithms for the $SINR_P$ Model

The computation of efficient schedules in the $SINR_P$ model has been studied in a more restricted number of papers. In [85], an efficient power-assignment algorithm, which schedules a strongly connected set of links in $O(\log^4 n)$ time slots in the $SINR_P$ model, is presented. In [48], randomized local algorithms for broadcasting are analyzed. The work of [18, 20, 58] proposes mathematical programming formulations for optimal schedules. However, the resulting formulations are infeasible from a computational point of view as the running time is exponential in the input.

3.2 Model and Definitions

We are interested in devising scheduling protocols that exhibit a provably good performance even in non-uniformly distributed networks. We therefore consider the network to consist of a set of n nodes $X = \{x_1, \ldots, x_n\}$ that are arbitrarily (possibly even worst-case) located in the Euclidean plane. The Euclidean distance between two nodes $x_i, x_j \in X$, is denoted by $d(x_i, x_j)$. For simplicity and without loss of generality, we assume that the minimal distance between any two nodes is 1.

A communication request l_i from a sender $s_i \in X$ to a receiver $r_i \in X$ is represented as a directed link (s_i, r_i) with length $d_i = d(s_i, r_i)$.

The Physical SINR Model ($SINR_P$)

A crucial aspect when studying scheduling in wireless networks is to use an appropriate model. In the past, researchers have studied a wide range of communication models, ranging from complex channel models to simplistic graph-based protocol models. A standard model that is realistic, but also concise enough to allow for stringent reasoning and proofs is the Physical *Signal-to-Interference-plus-Noise-Ratio* ($SINR_P$) model [53]. In this model, the successful reception of a transmission depends on the received signal strength, the interference caused by nodes transmitting simultaneously, and the ambient noise level. A message can be transmitted successfully if the ratio of the received signal strength and the sum of the interference caused by nodes sending simultaneously plus the noise level exceeds a hardware-dependent value β.

The received power $P_r(s_i)$ of a signal transmitted by sender s_i at an intended receiver r_i is

$$P_r(s_i) \;=\; P(s_i) \cdot g(s_i, r_i),$$

where $P(s_i)$ is the transmission power of s_i and $g(s_i, r_i)$ comprises the propagation attenuation (link gain) modeled as $g(s_i, r_i) = d_i^{-\alpha}$. The *path-loss exponent* α is a constant between 2 and 6, whose exact value depends on external conditions of the medium (humidity, obstacles, ...), as well as the exact sender-receiver distance. As common, we assume that $\alpha > 2$ [53].

Given a request $l_i = (s_i, r_i)$, we use the notation $I_r(s_j) = P_r(s_j)$ for any other sender s_j concurrent to s_i, in order to emphasize that the signal power transmitted by s_j is perceived at r_i as interference. The *total interference* I_r experienced by a receiver r_i is the sum of the interference power values created by all nodes in the network transmitting simultaneously (except the intending sender s_i), that is, $I_r := \sum_{s_j \in X \setminus \{s_i\}} I_r(s_j)$. Finally, let N denote the ambient noise power level. Then, r_i receives s_i's transmission if and only

if

$$\begin{aligned} SINR(i) &= \frac{P_r(s_i)}{N + \sum_{j \neq i} I_r(s_j)} \\ &= \frac{P(s_i)g(s_i, r_i)}{N + \sum_{j \neq i} P(s_j)g(s_j, r_i)} \\ &= \frac{\frac{P(s_i)}{d_i^\alpha}}{N + \sum_{j \neq i} \frac{P(s_j)}{d(s_j, r_i)^\alpha}} \geq \beta, \end{aligned} \tag{3.1}$$

where β is the minimum $SINR$ required for a successful message reception.

In the sequel we assume $\beta \geq 1$.

In this chapter we assume that all nodes transmit with the same power level. This assumption is also referred to as *uniform power assignment scheme* [52]. This kind of power assignment has been widely adopted in practical systems and has been studied in depth in [105].

For the sake of simplicity, in the following analysis sections, we set $N = 0$ and ignore the influence of noise in the calculation of $SINR$. However, this has no significant effect on the results.

Scheduling Problem

The aim of an algorithm for the Scheduling problem is to generate a short sequence of link sets, such that the $SINR$ level is above a threshold β at every intended receiver in each link set and all links are scheduled successfully at least once.

More precisely, let L be a set of communication requests. A *schedule* is represented by $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_T)$, where $\mathcal{S}_t$ denotes a subset of links of L, designated to time slot t. As in [53], we assume without loss of generality that transmissions are slotted into synchronized slots of equal length and in each time slot t, a node can either transmit or remain silent.

The task of a scheduling algorithm is to schedule a set of communication requests L such that all messages are *successfully* received.

Definition 3.1. *Consider a time slot t. The request $l_i = (s_i, r_i)$ is* successfully scheduled *in time slot t if r_i can decode message from s_i correctly according to the $SINR$ inequality* (3.1).

Let L_t be the set of all successfully scheduled links in time slot t. We aim at ensuring that after as few time slots as possible every link has been transmitted, i.e., the union of all sets L_t equals the set of requests L. The *scheduling complexity* defined in [85] is a measure that captures the amount of time required by a scheduling protocol to schedule requests in the $SINR_P$ model.

Definition 3.2. *The Scheduling problem for L consists in finding a schedule $\mathcal{S}$ of minimal length T such that the union of all successfully transmitted links $\bigcup_{t=1}^{T(\mathcal{S})} L_t$ equals L. An algorithm's* scheduling complexity *is the length of the schedule generated.*

Evidently, an algorithm's quality is reflected by its scheduling complexity. Ideally, a wireless scheduling protocol should achieve an optimal scheduling complexity in all networks and for arbitrary communication requests.

In the sequel, we assume that there are no conflicts in the transmission setup, i.e., each node is either a sender or a receiver and each receiver is associated with only one sender. These conflicts can be resolved efficiently by introducing additional nodes at the same position such that there is one sender-receiver pair for each link. Therefore we neglect them for simplicity's sake.

One-Shot Scheduling Problem

In contrast to the Scheduling problem, where we were interested in a schedule for all links, the objective of an algorithm solving the One-Shot Scheduling problem is to pick a subset of weighted links such that the total weight is maximized and the $SINR$ level is at least β at every scheduled receiver. In other words, we attempt to use one slot to its full capacity.

Formally, let L be a set of communication requests, where each link l_i is assigned a weight w_i. A set $\mathcal{S} = (l_1, l_2, \ldots, l_m) \subseteq L$ is a solution to an instance of a One-Shot Scheduling problem if the following two conditions hold:

$$\begin{aligned} \mathcal{S} &= \operatorname*{argmax}_{\mathcal{S}' \subseteq L} \sum_{l_j \in \mathcal{S}'} w_j, \\ SINR(r_j) &\geq \beta, \quad \forall l_j \in \mathcal{S}. \end{aligned}$$

3.3 NP-Hardness of Scheduling Problems

Solving problems in the $SINR$ setting is very difficult. Even finding an algorithm determining a good approximation for every problem instance is hard, as is documented by the vast amount of literature with heuristics on this subject [18, 20, 46, 50, 58, 85, 87].

As mentioned earlier in this chapter, there are hardly any results on the hardness of problems in a geometric setting. However, insights on the complexity are very important for the design of efficient algorithms. In this section we analyze the Scheduling problem and the One-Shot Scheduling problem and prove them to be NP-hard in the $SINR_P$ model.

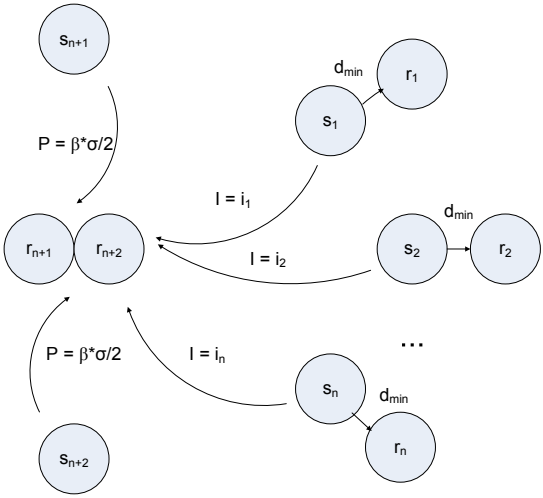


Figure 3.1: Reduction from Partition: link l_{n+1} (or l_{n+2}) can be scheduled if and only if the interference caused by simultaneously scheduled links $s_j, j \in \{1 \cdots n\}$ is less our equal to $\sigma/2$.

Scheduling Problem

Proving the Scheduling problem to be NP-hard implies that there exists no polynomial time algorithm for determining an optimal schedule, unless $P = NP$. It is widely believed that an NP-hard computational problem is not tractable efficiently.

To prove the hardness of the scheduling problem, we present a polynomial time reduction from the Partition problem, an NP-complete special case of the well known Subset Sum problem. If the solution to an instance of the Scheduling problem implies a solution to any instance of the Partition problem, Scheduling must be at least as hard as Partition.

Lemma 3.3. *The Partition problem is reducible to the Scheduling problem in polynomial time.*

Proof. The Partition problem (proved to be NP-complete by Karp in his seminal work [62]) can be formulated as follows: Given a set $\mathcal{I}$ of integers, is it possible to divide this set into two subsets $\mathcal{I}_1$ and $\mathcal{I}_2$, such that the sums of the numbers in each subset are equal? The subsets $\mathcal{I}_1$ and $\mathcal{I}_2$ must form a partition in the sense that they are disjoint and they cover $\mathcal{I}$.

Partition problem: Find $\mathcal{I}_1, \mathcal{I}_2 \subset \mathcal{I} = \{i_1, \ldots, i_n\}$ s.t.:

$$\begin{aligned} \mathcal{I}_1 \cap \mathcal{I}_2 &= \emptyset, \\ \mathcal{I}_1 \cup \mathcal{I}_2 &= \mathcal{I}, \quad \text{and} \\ \sum_{i_j \in \mathcal{I}_1} i_j &= \sum_{i_j \in \mathcal{I}_2} i_j = \frac{1}{2} \sum_{i_j \in \mathcal{I}} i_j. \end{aligned}$$

The proof proceeds as follows. First, we define a many-to-one reduction from any instance of the Partition problem to a geometric instance of the Scheduling problem. Then, we argue that the instance of the Scheduling problem cannot be scheduled in $T \leq 1$ time slots, but can be scheduled in $1 < T \leq 2$ time slots if and only if the instance of the Partition problem is solved.

Let us look at a set $\mathcal{I} = \{i_1, \ldots, i_n\}$ of integers, where the elements of $\mathcal{I}$ add up to σ,

$$\sum_{j=1}^{n} i_j = \sigma.$$

Without loss of generality, we can assume all elements to be distinct and positive. We construct the following Scheduling problem instance with $n+2$ links $L = \{l_1, \ldots, l_{n+2}\}$ (see Figure 3.1). We refer to the sender node belonging to l_j as s_j and the receiver node r_j. We assign each of these nodes a position (X,Y) in the plane. For each integer i_j in I we set the x-axis coordinate of s_j to $(P/i_j)^{1/\alpha}$,

$$pos(s_j) = \left(\left(\frac{P}{i_j} \right)^{\frac{1}{\alpha}}, 0 \right) \quad \forall\ 1 \leq j \leq n.$$

Next, we designate for every $r_i, 1 \leq i \leq n$ its position to be at distance $d_{\min}$ to its sender s_i, where

$$d_{\min} = P^{\frac{1}{\alpha}} \cdot \frac{\left(\frac{1}{(i_{\max}-1)^{1/\alpha}} - \frac{1}{i_{\max}^{1/\alpha}} \right)}{\left(1 + (n\beta)^{\frac{1}{\alpha}} \right)} \tag{3.2}$$

and $i_{\max}$ is the maximal value of the integers in set $\mathcal{I}$. Thus

$$pos(r_i) = pos(s_i) + (d_{\min}, 0).$$

Finally, we place r_{n+1} and r_{n+2} at the center $(0,0)$ and their senders s_{n+1}, s_{n+2} perpendicular to the x-axis, at distance $(2P/\beta\sigma)^{1/\alpha}$, i.e.,

$$\begin{aligned} pos(r_{n+1}) &= pos(r_{n+2}) = (0,0), \\ pos(s_{n+1}) &= \left(0, \left(\frac{2P}{\beta \cdot \sigma} \right)^{\frac{1}{\alpha}} \right), \\ pos(s_{n+2}) &= \left(0, -\left(\frac{2P}{\beta \cdot \sigma} \right)^{\frac{1}{\alpha}} \right). \end{aligned}$$

Having defined the geometric instance of the Scheduling problem for any instance of the Partition problem, we proceed by showing that in order to find a schedule of length $1 < T \leq 2$, a solution to the Partition problem is required. Clearly, it is not possible to schedule all links in one slot, since the receivers r_{n+1} and r_{n+2} are at the same position and we assume $\beta \geq 1$.

In order to transmit successfully, the $SINR$ constraint at the intended receiver has to be satisfied. In the following lemma we prove that the receivers $r_1, \ldots, r_n$ are close enough to their respective senders to guarantee successful transmission, regardless of the number of other links scheduled simultaneously.

Lemma 3.4. *Let $L_i = \{l_j | 1 \leq j \leq n+1$ and $i \neq j\}$. It holds for all $i \leq n$ that the $SINR$ exceeds β when the link l_i is scheduled concurrently with the set L_i,*

$$SINR(r_i) = \frac{\frac{P}{d_i^\alpha}}{\sum_{l_j \in L_i} \frac{P}{d(s_j, r_i)^\alpha}} > \beta.$$

We are not considering l_{n+2}, since l_{n+1} and l_{n+2} can never be scheduled simultaneously and the distance between s_{n+2} and any other node is the same as the distance between s_{n+1} and this node.

Proof. Since the positions of the sender nodes $s_1, \ldots, s_n$ depend on the values of $i_1, \ldots, i_n$, we can determine the minimum distance between two sender nodes s_j, s_k.

$$\begin{aligned} d(s_j, s_k) &= |d(s_j, r_{n+1}) - d(s_k, r_{n+1})| \\ &= \left| \left(\frac{P}{i_j}\right)^{\frac{1}{\alpha}} - \left(\frac{P}{i_k}\right)^{\frac{1}{\alpha}} \right| \\ &\geq P^{\frac{1}{\alpha}} \left(\frac{1}{(i_{\max} - 1)^{1/\alpha}} - \frac{1}{i_{\max}^{1/\alpha}} \right). \end{aligned} \tag{3.3}$$

Thus, one can deduce that the sender s_j closest to r_i, $i \neq j$ is located at least at distance $d(s_j, s_i) - d_{\min}$ from r_i ($d_{\min}$ is defined in (3.2)). All the other sender nodes (including s_{n+1}) are farther away. This suffices to show a lower bound for $SINR(r_i)$.

$$\begin{aligned} SINR(r_i) &> \frac{\frac{1}{d_{\min}^\alpha}}{\frac{n}{(d(s_j, s_i) - d_{\min})^\alpha}} \\ &\geq \frac{1}{n} \left(\left(1 + (n\beta)^{\frac{1}{\alpha}}\right) - 1 \right)^\alpha \\ &= \beta. \end{aligned} \tag{3.4}$$

□

Having proved that successful transmission is guaranteed for links $l_1, \dots l_n$, no matter how many other links are scheduled concurrently, we now return to the proof of Lemma 3.3.

We claim that there exists a solution to the Partition problem if and only if there exists a 2-slot schedule for L. For the first part of the claim, assume we know two subsets $\mathcal{I}_1, \mathcal{I}_2 \subset \mathcal{I}$, whose elements sum up to $\sigma/2$. To construct a 2-slot schedule, $\forall i_j \in \mathcal{I}_1$, we assign the link l_j to the first time slot, along with l_{n+1}, and assign the remaining links to the second time slot. Due to Lemma 3.4 we can focus our analysis on the receivers r_{n+1} and r_{n+2}. The situation is the same for both receivers, so it suffices to examine r_{n+1}. The signal power r_{n+1} receives from its sender node s_{n+1} is

$$P_{r_{n+1}}(s_{n+1}) = \frac{P}{\left(\left(\frac{2P}{\beta\sigma}\right)^{\frac{1}{\alpha}}\right)^{\alpha}} = \frac{\beta\sigma}{2}.$$

The interference r_{n+1} experiences from each sender s_j is

$$I_{r_{n+1}}(s_j) = \frac{P}{\left(\left(\frac{P}{i_j}\right)^{\frac{1}{\alpha}}\right)^{\alpha}} = i_j,$$

which results in total interference of

$$I_{r_{n+1}} = \sum_{i_j \in \mathcal{I}_1} i_j = \frac{\sigma}{2}.$$

This allows to lower bound the $SINR$ at r_{n+1}

$$SINR(r_{n+1}) \geq \frac{P_{r_{n+1}}(s_{n+1})}{I_{r_{n+1}}} = \frac{\beta\sigma/2}{\sigma/2} = \beta,$$

which, in combination with Lemma 3.4, proves that our schedule guarantees successful transmission for all links.

For the second part of the claim, we need to show that if no solution to the Partition problem exists, we cannot find a 2-slot schedule for L. No solution to the Partition problem implies that for every partition of $\mathcal{I}$ into two subsets, the sum of one set is greater than $\sigma/2$. Assume we could still find a schedule with only two slots. Since the receivers r_{n+1} and r_{n+2} are at the same position, they have to be assigned to different slots to permit a successful transmission. Because we have to split $L \setminus \{l_{n+1}, l_{n+2}\}$ into two sets and the received power from $s_j, j = 1, \dots, n$ at (0,0) is i_j, we end up with a total interference at (0,0) greater than $\sigma/2$ for one slot, which prevents the correct reception of the signal from s_{n+1} or s_{n+2}. □

By Lemma 3.3, Partition is reducible to Scheduling. Therefore, we can now state a theorem on the complexity of the Scheduling problem.

Theorem 3.5. *The Scheduling problem in the $SINR_P$ model is NP-hard.*

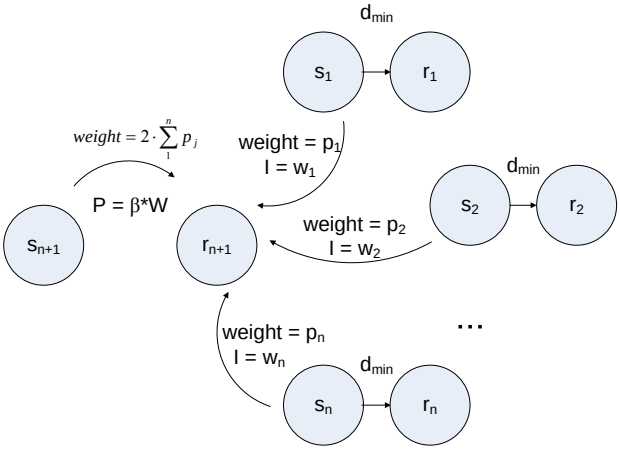


Figure 3.2: Reduction from Knapsack: the weight of simultaneously scheduled links is maximized if and only if the sum of the values p_j assigned to them is maximized and the knapsack capacity W is not violated.

One-Shot Scheduling problem

In this section we prove that the decision version of the weighted One-Shot version of the Scheduling problem, under uniform power assignment scheme, is also NP-hard in the $SINR_P$ model. We proceed by describing a polynomial time reduction for the Knapsack problem in Lemma 3.6.

Lemma 3.6. *Knapsack is reducible to the One-Shot Scheduling problem in polynomial time.*

Proof. Let us first introduce the Knapsack problem: Consider n kinds of items, x_1 through x_n, where each item x_j has a value p_j and a weight w_j. The maximum weight that we can carry in a bag is W. Our aim is to choose the items we put in the bag such that the sum of the values is maximized. We can formulated this task as an integer program.
Knapsack problem:

$$\max \sum_{j=1}^{n} p_j x_j, \qquad \text{s.t.} \tag{3.5}$$

$$\sum_{j=1}^{n} w_j x_j \quad \leq \quad W, \tag{3.6}$$

$$x_j \quad \in \quad \{0,1\}, \quad j = 1, \ldots, n$$

Without loss of generality, we assume that there are only items of distinct integer weights. As in the proof for the Scheduling problem, we start by defining a many-to-one reduction from any instance of the Knapsack problem to a geometric instance of the One-Shot Scheduling problem, and afterwards prove that the latter can be solved if and only if the former is also solved.

We have to dispose links in the plane, such that the rules of the Knapsack problem are enforced (see Figure 3.2). We position a sender node s_i in the plane for each x_i, such that the received power from s_i at (0,0) is w_i, i.e.,

$$pos(s_i) = \left(\left(\frac{P}{w_i} \right)^{\frac{1}{\alpha}}, 0 \right), \quad \forall 1 \leq j \leq n.$$

Now we set r_i close enough to s_i to guarantee successful reception regardless of other links.

$$pos(r_i) = pos(s_i) + (d_{min}, 0), \quad \text{where}$$

$$d_{min} = P^{\frac{1}{\alpha}} \cdot \frac{\left(\frac{1}{(w_{\max}-1)^{1/\alpha}} - \frac{1}{w_{\max}^{1/\alpha}} \right)}{\left(1 + (n\beta)^{\frac{1}{\alpha}} \right)},$$

and $w_{\max}$ is the largest weight in this problem instance.

In the next step we place an additional link l_{n+1}, such that r_{n+1} is at (0,0) and s_{n+1} is in such a distance that the received power at (0,0) is βW.

$$\begin{aligned} pos(r_{n+1}) &= (0,0), \\ pos(s_{n+1}) &= \left(0, \left(\frac{P}{\beta W} \right)^{\frac{1}{\alpha}} \right). \end{aligned}$$

Thereafter, we assign a weight to each link:

$$\begin{aligned} weight(l_i) &= p_i, \quad \forall 1 \leq i \leq n \\ weight(l_{n+1}) &= 2 \cdot \sum_{j=1}^{n} p_j. \end{aligned}$$

Note that $SINR(r_i) > \beta, \forall i = 1 \ldots n$, even if all link transmissions are concurrent, since we can apply Lemma 3.4 (due to the fact that we chose the distance between a sender and a receiver of a link to be $d_{\min}$ in both reductions). If we execute an algorithm solving this One-Shot Scheduling problem, we obtain a solution for the Knapsack problem: Let $\mathcal{S}_{OPT}$ be the set of links of an optimal solution to the One-Shot problem constructed above. The described assignment of weights ensures that l_{n+1} is picked,

since without it the maximal sum of weights cannot be reached. We can compute $SINR(r_{n+1})$ as follows

$$\begin{aligned} SINR(r_{n+1}) &= \frac{P_{r_{n+1}}(s_{n+1})}{I_{r_{n+1}}} \\ &= \frac{\frac{P}{\left(\left(\frac{P}{\beta W}\right)^{\frac{1}{\alpha}}\right)^{\alpha}}}{\sum_{l_j \in \mathcal{S}_{OPT}} \frac{P}{\left(\left(\frac{P}{w_j}\right)^{\frac{1}{\alpha}}\right)^{\alpha}}} \\ &= \beta \cdot \frac{W}{\sum_{l_j \in \mathcal{S}_{OPT}} w_j}, \end{aligned}$$

and since a valid solution allows l_{n+1} to be transmitted successfully, we have $SINR(r_{n+1}) > \beta$. Consequently a solution to the One-Shot Scheduling problem satisfies

$$\sum_{l_j \in \mathcal{S}_{OPT}} w_j < W.$$

Hence, each of the selected links l_i stands for x_i in (3.5) and (3.6), which fulfills the condition of the Knapsack problem. Because $\mathcal{S}_{OPT}$ maximizes the sum of the weights at the same time, the sum of the values of the items of the Knapsack problem is maximized as well. □

Lemma 3.6 implies that no algorithm can solve the One-Shot Scheduling problem without solving an NP-hard problem. Thus the One-Shot Scheduling problem is NP-hard as well.

Theorem 3.7. *One-Shot Scheduling in the $SINR_P$ model is NP-hard.*

In contrast to these results on the complexity of scheduling with a *uniform power assignment*, the question whether the Scheduling problem *with* power control is also NP-hard remains open and is an area of active research.

3.4 Approximation Algorithms

In this section we propose two approximation algorithms for the Scheduling and the One-Shot Scheduling problems.

Before describing the algorithms, let us introduce the notion of *length diversity*, namely the number of magnitudes of distances. Formally, $g(L)$ is defined as

$$g(L) := |\{m | \exists l_i, l_j \in L : \lfloor \log(d_i/d_j) \rfloor = m\}|. \tag{3.7}$$

For our problem, $g(L)$ denotes the number of non-empty length classes of the set of links to be scheduled. In realistic scenarios, the diversity $g(L)$ is usually a small constant.

The algorithms we present consist of two steps: First, the problem instance is partitioned into disjoint link length classes; then, a feasible schedule is constructed for each length class using a greedy strategy.

Scheduling

Algorithm 3.1 Approximation Algorithm for the Scheduling problem

Input: A set L of links located arbitrarily in the Euclidean plane
Output: A schedule $\mathcal{S}$ in which every link can be transmitted successfully

1: Let $R = R_0, \ldots, R_{\log(l_{\max})}$ such that R_k is the set of links l_i of length $2^k \leq d_i < 2^{k+1}$;
2: $t = 1$;
3: **for all** $R_k \neq \emptyset$ **do**
4: Partition the plane into squares of width $\mu \cdot 2^k$;
5: 4-color the cells such that no two adjacent cells have the same color.
6: **for** $j = 1$ **to** 4 **do**
7: Select color j;
8: **repeat**
9: For each square A of color j, pick one link $l_i \in R_k$ with receiver r_i in A, assign it to time slot t ($L_j^k = L_j^k \cup l_i$);
10: $t = t + 1$; $\mathcal{S}_t = L_j^k$;
11: **until** all links of R_k in the selected squares are scheduled
12: **od**
13: **od**
14: **return** $\mathcal{S}$;

The algorithm (for a description in pseudo-code see Algorithm 3.1) starts by partitioning the input set of links L into length classes $(R_0, \cdots, R_{g(L)})$. Each subset R_k is scheduled separately. First, the plane is partitioned into square grid cells of side $\mu \cdot 2^k$, where μ is defined as follows

$$\mu = 4 \left(8\beta \cdot \frac{(\alpha - 1)}{(\alpha - 2)} \right)^{\frac{1}{\alpha}}, \tag{3.8}$$

and then the cells are colored regularly with 4 colors (see Figure 3.3). Links whose receivers belong to different cells of the same color are scheduled simultaneously (added to set L_j^k). Note that the inner *repeat* loop (lines 8-11) constructs a schedule of length $\Delta(A_{max}^k)$, which is the maximum number of links in length class k, whose receivers are in the same grid cell A^k. Given that there are 4 colors and $g(L)$ length classes, all links a scheduled in $4 \cdot \Delta(A_{max}^k) \cdot g(L)$ time slots.

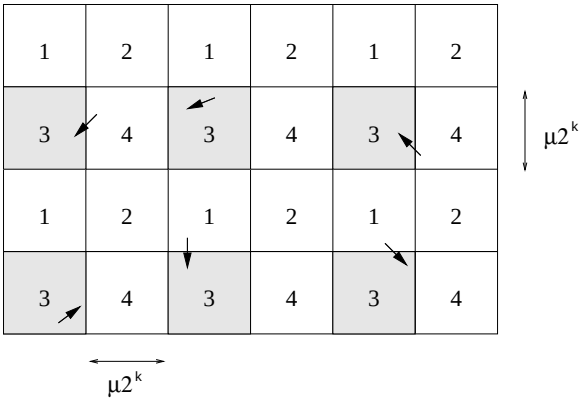


Figure 3.3: In line 7 of Algorithm 1, the algorithm picks all squares numbered by j. The example shows an inner loop iteration for length class R_k and $j = 3$. The algorithm schedules one unscheduled link from each selected square (if there exists one).

We show now that the schedule obtained by Algorithm 3.1 is correct, by proving in Theorem 3.8 that all links can be scheduled successfully in their respective time slot.

Theorem 3.8. *Consider an arbitrary set of links L to be scheduled. For every time slot t, the set $\mathcal{S}_t$ of links output by Algorithm 3.1 is scheduled successfully, i.e., the $SINR$ at every intended receiver is larger than β.*

Proof. We demonstrate that all transmissions scheduled in a time slot t are received successfully by the intended receivers, i.e., their $SINR$ is sufficiently high.

Without loss of generality, let us examine links in a length class R_k. Every link $l_i \in R_k$ satisfies $d_i < 2^{k+1}$, thus the perceived power at r_i from s_i is at least

$$P_{r_i}(s_i) \geq \frac{P}{2^{\alpha(k+1)}}. \tag{3.9}$$

Since Algorithm 3.1 schedules at most one link in each cell with the same color concurrently, the closest 8 senders s_j scheduled in the same time slot must be at least at distance $d(r_i, s_j) \geq \mu 2^k - 2^{k+1} = 2^k(\mu - 2)$ to r_i (see Figure 3.3). Consequently, the sum of their interference experienced by r_i is

less than

$$\sum_{j=1}^{8} P_{r_i}(s_j) \leq \frac{8P}{(2^k(\mu-2))^\alpha}.$$

In the next step, we consider the (at most) 16 senders s_j at distance $3\mu 2^k - 2^{k+1} \leq d(r_i, s_j) \leq 5\mu 2^k - 2^{k+1}$. They contribute a total interference of

$$\sum_{j=9}^{25} P_{r_i}(s_j) \leq \frac{16P}{(2^k(3\mu-2))^\alpha}.$$

We continue aggregating the interference from nodes s_j at distance range

$$(2l-1)\mu 2^k - 2^{k+1} \leq d(r_i, s_j) < (2l+1)\mu 2^k - 2^{k+1},$$

$\forall l = 1, 2, \ldots$. Since at most $8l$ links are picked in each interval, the interference caused by them is at most

$$\sum_{\substack{d(r_i,s_j)< \\ (2l+1)\mu 2^k - 2^{k+1} \\ d(r_i,s_j)\geq \\ (2l-1)\mu 2^k - 2^{k+1}}} P_{r_i}(s_j) \leq \frac{8P \cdot l}{(2^k((2l-1)\mu-2))^\alpha}.$$

Thus, the total interference at a scheduled receiver r_i can be upper bounded by

$$\begin{aligned}
I_{r_i} &\leq \sum_{l=1}^{\infty} \frac{8P \cdot l}{(2^k((2l-1)\mu-2))^\alpha} \\
&\leq \frac{8P}{2^{k\alpha}} \sum_{l=1}^{\infty} \frac{l}{(\frac{1}{2}(2l-1)\mu)^\alpha} \qquad (3.10) \\
&\leq \frac{8P}{2^{(k-1)\alpha}\mu^\alpha} \sum_{l=1}^{\infty} \frac{l}{(2l-l)^\alpha} \\
&\leq \frac{8P}{2^{(k-1)\alpha}\mu^\alpha} \sum_{l=1}^{\infty} \frac{1}{l^{\alpha-1}} \\
&\leq \frac{8P}{2^{(k-1)\alpha}\mu^\alpha} \frac{(\alpha-1)}{(\alpha-2)}, \qquad (3.11)
\end{aligned}$$

where (3.10) follows because $x - 2 > x/2, \quad \forall x > 4$ and $\mu > 4$, given that $\beta \geq 1$ and $\alpha \geq 2$; and (3.11) follows from a bound on Riemann's zeta function. Using (3.9), (3.11), and plugging in the value of μ, defined in (3.8), the $SINR$ at receiver r_i can be lower bounded by

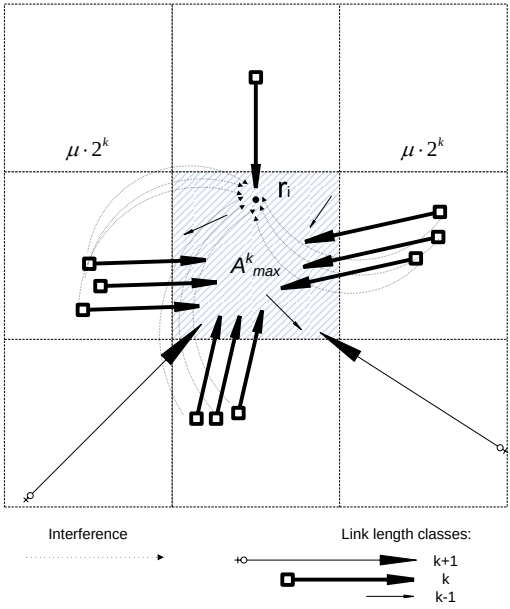


Figure 3.4: Lower Bound: an optimum algorithm could schedule at most q links with receivers in A^k_{max} in length class k in a single time slot.

$$\begin{aligned} SINR(r_i) &= \frac{P_{r_i}(s_i)}{I_{r_i}} \\ &> \frac{\frac{P}{2^{\alpha(k+1)}}}{\frac{8P}{2^{(k-1)\alpha}\mu^\alpha}\frac{(\alpha-1)}{(\alpha-2)}} \\ &= \frac{\mu^\alpha}{4^\alpha \cdot 8 \cdot \frac{(\alpha-1)}{(\alpha-2)}} \\ &= \beta, \end{aligned}$$

□

Now we turn our attention to the efficiency of Algorithm 3.1. In particular, in Theorem 3.9 we bound its approximation ratio.

Theorem 3.9. *The approximation ratio of Algorithm 3.1 is $O(g(L))$, where $g(L)$ is the length diversity of the input, defined in (3.7).*

Proof. The proof relies on the choice of a so called *critical* square $A^k_{max} = \mu 2^k \times \mu 2^k$ (see Figure 3.4), i.e., we choose the cell with the highest density $\Delta(A^k_{max})$ over all $g(L)$ generated grids. Note that $\Delta(A^k_{max})$ is the number of links l_i whose receiver is located in cell A^k_{max} and whose length class is k, i.e., $2^k \leq d_i < 2^{k+1}$. We proceed by showing that an optimum algorithm OPT can schedule all $\Delta(A^k_{max})$ in at least $T_{OPT} = \lceil \Delta(A^k_{max})/q \rceil$ time slots, where q is a constant dependent on parameters α and β (μ is defined in (3.8)):

$$q = \frac{\left(2(\sqrt{2}\mu + 1)\right)^{\alpha}}{\beta}. \tag{3.12}$$

Assume, by contradiction, that OPT schedules all links in less than T_{OPT} time slots. Therefore, there must exist a time slot $t', 1 \leq t' \leq T_{OPT}$, such that more than q links in A^k_{max} are scheduled simultaneously. We pick one of the scheduled links $l_i, r_i \in A^k_{max}$ in time slot t' and calculate the resulting $SINR$ level at r_i:

$$\begin{aligned} SINR(r_i \in A^k_{max}) &\leq \frac{\frac{P}{d_i^{\alpha}}}{P \cdot \sum_{j=0}^{q} d(s_j, r_i)^{-\alpha}} \\ &< \frac{\frac{P}{2^{k\alpha}}}{P \cdot q \cdot (2\sqrt{2}\mu 2^k + 2^{k+1})^{-\alpha}} \qquad (3.13) \\ &= \beta, \qquad (3.14) \end{aligned}$$

where (3.13) follows from the fact that $d_i \geq 2^k$, $d_j < 2^{k+1}$ and $d(r_i, r_j) \leq 2\sqrt{2}\mu 2^k$; and (3.14) follows from definition (3.12) of q.

Hence, to schedule all links in the *critical* square A^k_{max}, OPT needs time

$$T_{OPT} \geq \left\lceil \frac{\Delta(A^k_{max})}{q} \right\rceil. \tag{3.15}$$

On the other hand, Algorithm 3.1 schedules all links in L in time

$$T(Algorithm\ 3.1) \leq 4 \cdot \Delta(A^k_{max}) \cdot g(L). \tag{3.16}$$

The approximation ratio follows from (3.15) and (3.16):

$$\begin{aligned} \frac{T(Algorithm\ 3.1)}{T_{OPT}} &\leq 4q \cdot g(L) \\ &= O(g(L)). \qquad (3.17) \end{aligned}$$

□

One-Shot Scheduling

Algorithm 3.1 can be adapted to solve the weighted One-Shot Scheduling problem described in Section 3.2 (see pseudo code in Algorithm 3.2). As before, the input set L is partitioned into $g(L)$ length classes, and grids with cell size $\mu \cdot 2^k, k \in \{0 \cdots g(L)\}$ are colored with 4 colors $j \in \{1 \cdots 4\}$. Then, $4 \cdot g(L)$ feasible schedules L_j^k are generated by greedily picking the *heaviest* link in each square A^k of the same color. In the end, the heaviest set of links among all colors and all link classes is chosen.

Algorithm 3.2 Approximation Algorithm for One-Shot Scheduling

Input: A set L of links located arbitrarily in the Euclidean plane
Output: A subset L_j^k in which every link can be transmitted successfully and the total weight $w(L_j^k)$ is maximized

1: Let $R = R_0, \ldots, R_{\log(l_{\max})}$ such that R_k is the set of links l_i of length $2^k \leq d_i < 2^{k+1}$;
2: **for all** $R_k \neq \emptyset$ **do**
3: Partition the plane into squares of width $\mu \cdot 2^k$;
4: 4-color the cells such that no two adjacent cells have the same color.
5: **for** $j = 1$ **to** 4 **do**
6: For each square A of color j, pick the *heaviest* link $l_i \in R_k$ with receiver r_i in A, assign it to L_j^k ($L_j^k = L_j^k \cup l_i$);
7: **od**
8: **od**
9: **return** $\text{argmax}_{L_j^k} \sum_{l_i \in L_j^k} w(l_i)$;

Since we pick one link per selected square, the feasibility of any schedule L_j^k constructed by Algorithm 3.2 has been proved in Theorem 3.8. In the next theorem we analyze the approximation ratio of this algorithm.

Theorem 3.10. *The approximation ratio of Algorithm 3.2 is $O(g(L))$, where $g(L)$ is the length diversity of the input (defined in (3.7)).*

Proof. We start by defining OPT_k to be a subset of the optimum schedule OPT comprised by links that belong to length class k, i.e., $2^k \leq d_i \in OPT_k < 2^{k+1}$. Observe that

$$w(OPT) = \sum_{k=0}^{g(L)} w(OPT_k). \tag{3.18}$$

In Theorem 3.9 we showed that an optimum algorithm could schedule at most q (defined in (3.12)) links in each cell A_k at a time. Therefore, given that every feasible schedule L_j^k computed by Algorithm 3.2 contains the heaviest

link in every forth cell, the following bound holds:

$$w(L_j^k) \geq \frac{1}{4q} \cdot w(OPT_k), \quad \forall j \in \{1 \cdots 4\}, k \in \{0 \cdots g(L)\}. \tag{3.19}$$

Since Algorithm 3.2 returns the schedule L_j^k of maximum weight over all length classes and colorings (there are at most $4 \cdot g(L)$ schedules L_j^k), the approximation ratio follows:

$$\begin{aligned} \operatorname*{argmax}_{L_j^k} w(L_j^k) &\geq \frac{1}{4 \cdot g(L)} \cdot \sum_{k=0}^{g(L)} w(L_j^k) \\ &\overset{(3.19)}{\geq} \frac{1}{16q \cdot g(L)} \cdot \sum_{k=0}^{g(L)} w(OPT_k) \\ &\overset{(3.18)}{=} \frac{w(OPT)}{16q \cdot g(L)} \end{aligned} \tag{3.20}$$

$$\Rightarrow \quad \begin{aligned} \frac{w(OPT)}{w(Algorithm\ 3.2)} &\leq 16q \cdot g(L) \\ &= O(g(L)). \end{aligned} \tag{3.21}$$

□

Because of ambient noise, there is usually a maximal distance for a successful transmission in realistic scenarios. Moreover, because of hardware size, a sender and a receiver cannot be arbitrarily close to each other. Hence, one can establish constant minimum and maximum link lengths, which results in a constant number of link length classes $g(L)$. Using this observation, we can state the following corollary.

Corollary 3.11. *Assuming a constant maximum and minimum link length, $g(L)$ is constant, and Algorithms 3.1 and 3.2 achieve constant approximation ratios.*

3.5 Concluding Remarks

In this chapter we wanted to gain deeper insights into the complexity of scheduling in wireless ad-hoc networks. To the best of our knowledge, we presented the first NP-hardness proofs for the $SINR_P$ model. As opposed to other NP-hardness proofs proposed for wireless networks, which rely on a graph structure and an arbitrary gain matrix, our proof explores the geometric nature of such networks – a property, which we consider fundamental.

When the distribution of nodes on the Euclidean plane is considered, all the entries in the gain matrix become constrained by the other entries. Therefore, arguing that two nodes cannot transmit concurrently in a schedule becomes much harder. Hence, a different kind of proof is necessary, where the constraints of the NP-complete problem are enforced by the geometry of the reduction.

4

Power Control and Scheduling

In the previous chapter, we assumed that every node transmitts with the same power level. Depending on the hardware, nodes are able to adjust their transmission power. This capability can increase the number of links that can transmit succesfully at the same time. To exploit this fact, we need efficient power control and scheduling algorithms that assign a power level to each node for every time slot. However, even an optimal power control algorithm cannot guarantee acceptable $SINR$ levels for all links concurrently because, in general, only a subset of all links can be scheduled in parallel. It is therefore unavoidable to postpone the transmission of some communication requests to subsequent time slots. As short schedules maximize network throughput, the ultimate aim of any scheduling algorithm remains hence to find a schedule of minimum (or at least close to minimal) length.

4.1 Simulations vs Worst Case Analysis

Scheduling and power control being of utmost theoretical and practical importance for wireless networks, it is not surprising that numerous algorithms are known for this problem. Most of them have in common that they were developed using heuristic (rule of thumb) reasoning, and evaluated through complex simulations. It is clear that simulation is problematic, as one can never cover all possible scenarios. What if an algorithm works well in most (simulated) scenarios but is inferior in some other classes of scenarios? What if these devastating classes are important or even critical in practice? In contrast, analytic worst-case analysis has the advantage to include all possible cases, and offers strict performance guarantees. In this chapter, want to overcome this shortcome and thoroughly analyze the existing algorithms. We propose a new measure *disturbance* in order to comprise the intrinsic difficulty of finding a short schedule for a problem instance. Moreover, we prove that for certain problem instances all existing algorithms we are aware

of perform poorly even if disturbance is low. In addition we present a new scheduling algorithm called LDS that exhibits explicit worst-case guarantees. We prove that on a class of scenarios our algorithm performs *exponentially* better than the previous algorithms. In particular, LDS schedules n transmission requests in $O(\log^2 n)$ time opposed to previous algorithms requiring $\Omega(n)$ time.

Overview of Existing Algorithms

A wide range of models and various classes of algorithms have been suggested in order to solve the problem of scheduling and power control. As mentioned in the previous chapter, scheduling algorithms defined for a graph-based interference model are of inferior quality in practices since they model interference as a binary property and ignore the accumulated interference of a large number of distant nodes. Moreover, graph-based scheduling algorithms are too conservative as they do not tap the full potential of spatial reuse. Overlapping links, for instance, are not scheduled simultaneously in a graph-based scheduling algorithm, although this is feasible in pratice [85].

As we argue in detail in Section 4.3, algorithms explicitly defined for the $SINR_P$ model can broadly be classified into three categories. One approach is to assign the same power to all transmitting nodes. In [85] it is shown that algorithms with such uniform power assignment can result in long schedules. In the same paper it is proved that the second intuitive procedure, adjusting the power proportionally to the so-called "energy-metric", can lead to long schedules as well.

More sophisticated methods are based on results from [1], where Aein shows how to determine the *maximum achievable* $SINR^*$ in polynomial time for satellite communications system. These results being directly applicable to wireless networks, it is possible to find an optimal power assignment efficiently. However, the problem is that $SINR^*$ may be too low to guarantee correct reception at all receivers. That implies that our problem of partitioning the set of communication requests into time slots meeting the required $SINR$ criteria remains unresolved. A brute force approach for finding the optimal schedule attempts to find for each time slot the largest set of remaining links which can be scheduled simultaneously by checking for each subset of links whether it allows a sufficiently high $SINR$. As there are 2^n subsets, however, the required time complexity grows exponentially with the number of links.

Consequently, many computationally efficient methods for postponing the transmission of links according to some ("link removal") heuristics have been devised. The first among them is presented by Zander [118]. He proposes an algorithm called SRA, which removes nodes from the current time slot by a stepwise approximation criterion, involving row and column sums of the gain matrix. He examines the problem in a model for cellular networks, but his

results can be adapted to other wireless networks easily. He demonstrates that SRA returns better results than algorithms with fixed power levels or constant received power levels with simulations. In [119] Zander devises an improved algorithm called LISRA that requires less knowledge of the network and the $SINR$ for each time slot converges to $SINR^*$ in a distributed fashion without knowing all link gains. Subsequently, several improvements on this convergence procedure have been proposed. E.g., Grandhi et al. [49] devised a distributed algorithm that converges exponentially fast to the maximum $SINR$ level which is achievable for a certain system. A similar approach is pursued in [44], along with the description of an extensive simulation. In these papers, the aim is to guarantee fast convergence for the power vector, a goal not considered in our work, where we focus on power control in combination with scheduling. The idea of Lee et al. [74] is to postpone links which either have a high level of interference at the receiver or links of which the sender causes much interference to other receivers. They compared their and Zander's algorithms with a simulation where the nodes are distributed uniformly at random. The distributed algorithm proposed by Wang et al. [110] eliminates links which cause most interference in order to allow the remaining links to reach an acceptable $SINR$ level. They present simulation results to show that their algorithm produces schedules close to the global optimum solution, when the nodes and links are distributed uniformly at random. In contrast to other authors they include noise, the system processing gain and the maximum possible power level in their model. Most recently, Brar et al. [23] present a scheduling method that is based on a greedy assignment of weighted colors.

All the above polynomial-time algorithms have one crucial drawback: The authors provide no worst-case analysis on their performance and all assumptions on their algorithm's quality are based on simulations and—in the case of [23]—analysis of randomly deployed networks. In Section 4.3, we show that these link removal heuristics have a bad worst-case performance, creating schedules which are exponentially longer than necessary for certain networks.

The same limitation holds for the influential algorithm for next neighbor transmissions and power control by ElBatt and Ephremides [38]. They combine two heuristics to produce a short schedule and the corresponding power assignment. First a set of *valid* links is selected by greedily choosing nodes such that no node is receiving and transmitting in the same slot (to avoid self-interference) and no sender is situated within a certain range of an already selected receiving node. In a second phase, Zander's LISRA algorithm is applied to these links. As it is possible to construct scenarios, where all links together form a valid set, the worst case behavior of LISRA carries over to the algorithm of [38] as well (see Section 4.3).

Recently, polynomial-time algorithms with provable guarantees in

$SINR_P$ model environments for specific network topologies were proposed and analyzed in [85, 87]. In this chapter, we improve on these algorithms and give strict worst-case guarantees even in scenarios in which no efficient bounds have been derived.

Other aspects of scheduling and power control are studied for instance in [20, 32, 93, 96, 97].

4.2 Model and Definitions

We adopt the same model as in Chapter 3 with two exceptions. We assume that the nodes can adjust their transmission power and we include noise in the description and in the analysis of our algorithm.

Problem Formulation

The aim of a scheduling and power control algorithm is to generate a sequence of power assignment vectors, such that the $SINR$ level is above a threshold β at every intended receiver and all links are scheduled successfully at least once.

More formally, let L be a set of *communication requests* l_i. P_t denotes the *power assignment vector*, where $P_t(s_i)$ determines the transmission power of sender s_i in time slot t. A *schedule* is represented by $\mathcal{S} = (P_1, \ldots, P_T)$. The task of a scheduling algorithm is construct a schedule such that all messages are *successfully* received. As in the previous chapter, we measure the quality of an algorithm by its scheduling complexity, the length of the schedules it generates.

Feasibility

Remember that in the $SINR_P$ model, the propagation attenuation (or link gain) between a sender node s_i and a receiver node r_j in the Euclidean plane is modeled as $g(s_i, r_j) = d(s_i, r_j)^{-\alpha}$. Whether or not a set of links can be scheduled in the same time slot depends on the link gain between all sender/receiver pairs. Note that the link gain matrix

$$Z = \left[\frac{g(s_j, r_i)}{g(s_i, r_i)}\right]_{i,j} = \left[\frac{d_i^\alpha}{d(s_j, r_i)^\alpha}\right]_{i,j}$$

is a matrix consisting of positive values only. Zander showed in [119], that this property can be exploited to compute the *maximum achievable* SIR^* for wireless networks efficiently.[1]

[1] [119] ignores the influence of noise, thus his results are valid in the *Signal-to-Interference-Ratio* (SIR) model. See [95] for an approach that handles noise as well.

Finding a power assignment yielding the maximal SIR level can essentially be reduced to solving an Eigenvalue problem for the link gain matrix Z. For positive matrices there is exactly one real eigenvalue λ^* for which all elements of the corresponding eigenvector have the same sign. Zander showed that the maximum achievable SIR^* is given by

$$SIR^* = \frac{1}{\lambda^* - 1}.$$

Furthermore, the corresponding eigenvector $\mathbf{P}^*$ constitutes a power vector reaching this maximum for all links, i.e. they all have the same SIR level.

Theorem 4.1 (Zander)**.** *In the absence of noise, a set of links can be successfully scheduled in one time slot if and only if the largest eigenvalue of the link gain matrix is less than* $1/\beta + 1$.

Inevitably there is noise in every real system. Nonetheless, this result is still useful since it provides a efficient method to determine if a set of links can NOT be scheduled concurrently. The converse however does not hold in the $SINR_P$ model.

Disturbance

Since we study arbitrary, possibly worst-case network and request settings, we introduce a formal measure that captures the intrinsic difficulty of scheduling a given set of communication requests.

For a given set of communication requests L and some constant $\rho \geq 1$, we define the *ρ-disturbance* as the maximal number of senders (receivers) that are in close physical proximity (depending on the parameter ρ) of any sender (receiver). Consider disks S_i and R_i of radius d_i/ρ around sender s_i and receiver r_i, respectively. Formally, the *ρ-disturbance of a link l_i* is the larger of either the number of senders in S_i or the number of receivers in R_i. The *ρ-disturbance of L* is then the maximum ρ-disturbance of any link $l_i \in L$.

Definition 4.2. *Given a set of requests L, the ρ-disturbance, denoted as χ_ρ of L is defined as*

$$\chi_\rho := \max_{l_i \in L} \chi_\rho(l_i),$$

where the disturbance $\chi_\rho(l_i)$ for request l_i is the maximum of $|\{r_j \mid d(r_j, r_i) \leq d_i/\rho\}|$ and $|\{s_j \mid d(s_j, s_i) \leq d_i/\rho\}|$.

As it turns out, the *disturbance* of a set of requests indeed captures the fundamental difficulty of scheduling these requests. Solving problem instances with low disturbance efficiently is very important in practice since in realistic networks one always tries to prevent situations with many receivers clustered in the same area. Section 4.4 presents LDS, a scheduling algorithm that achieves a provably fast performance for all networks and requests that

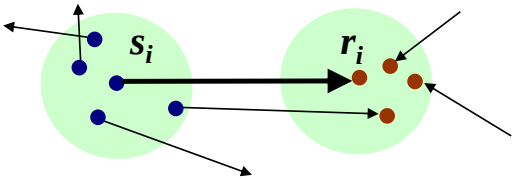


Figure 4.1: Illustrating example for the disturbance of a communication request l_i. For $\rho = 3$ the disturbance in this case is $\chi_3(l_i) = 4$.

have low disturbance. On the other hand, we prove in Section 4.3 that currently known scheduling algorithms may perform highly sub-optimally even in instances with low disturbance. In fact, the number of time slots required by any such algorithm may be exponentially higher than the optimum.

4.3 Inefficiency of Existing Protocols

Classification of Power Control Algorithms

Intuitively, the disturbance of a set of requests in a network characterizes the difficulty of scheduling these requests in a wireless communication environment. Therefore, an efficient scheduling algorithm should be capable of generating short schedules in settings with low disturbance. Unfortunately, all previously known scheduling algorithms may require a linear number of time slots in order to schedule a set of requests even if their ρ-disturbance is as low as 1.

Existing scheduling algorithms for the $SINR_P$ model can be classified into three classes:[2]:

- *uniform power assignment*: the transmission power of all nodes is the same.
- *linear power assignment*: the transmission power for a link of length d_i is set to a value proportional to d_i^α. Protocols analyzed using the so-called "energy-metric" belong to this category.
- *link removal heuristics*

Uniform and Linear Power Assignment

Recently, it has been proven in [85] that every algorithm employing a *uniform or linear power assignment* scheme has a poor worst-case efficiency. In

[2]Notice that algorithms based on graph-models can typically be characterized as either employing a uniform or linear power assignment scheme.

particular, any such algorithm may require a linear number of time slots even if every node merely wants to transmit to its closest neighbor in the network.

Theorem 4.3 ([85]). *Every algorithm employing a uniform or linear power assignment scheme has a worst-case scheduling complexity of* $\Omega(n)$ *even in settings with* ρ*-disturbance 1.*

Theorem 4.3 indicates that a large number of scheduling algorithms proposed in the literature has bad worst-case behavior.

Link Removal Algorithms

In contrast to these intuitive, but inefficient scheduling schemes, *link removal heuristics* are much more sophisticated. The heuristics known in the literature are all based on a generic link removal algorithm.

Algorithm 4.1 Generic Link Removal Algorithm

```
1: time slot t := 1;
2: while there are links to schedule do
3:    compute SINR* and P* from Z;
4:    while SINR* ≤ β do
5:       remove links l_k for which CON is satisfied;
6:       compute SINR* and P* from new Z;
7:    od
8:    schedule the links of Z in time slot t and assign P*;
9:    time slot t := t + 1;
10:   compute new Z for unscheduled links;
11: od
```

The idea of these algorithms is to postpone the transmission of a link l_k from the set of the links if some condition CON holds, until the minimal $SINR$ level for successful reception is met. Then the optimal power vector is assigned and the procedure is repeated with the remaining links.

We scrutinize the four algorithms $SRA, SMIRA, WCRP$ and $LISRA$, which follow the execution of the generic algorithm and differ only in the condition CON.

SRA (Stepwise Removal Algorithm), devised by Zander in [118], iteratively removes the link with the largest row or column sum of Z, since these sums provide a bound on the maximal eigenvalue, until the required $SINR$ level is met.

$$CON : \max\{\sum_j Z_{kj}, \sum_j Z_{jk}\} \textit{ is maximimal for } k.$$

SMIRA (Stepwise Maximum Interference Removal Algorithm), by Lee et al. [74], excludes links which cause or receive the most interference when power is assigned optimally, taking the normalized link gain matrix Z and the corresponding optimal power vector into account.

$$CON : \max\{\sum_{j\neq k} P_j Z_{kj}, P_k \sum_{j\neq k} Z_{jk}\} \; is \; maximimal \; for \; k.$$

Lee et al. suggest versions of this algorithm considering only $\max_k(\sum_{j\neq k} P_j Z_{kj})$ or $\max_k(P_k \sum_{j\neq k} Z_{jk})$ in the condition and demonstrate with simulations, that they perform worse than SMIRA. Our analysis can be adapted easily to these cases with the same complexity result.

WCRP is a (distributed) algorithm presented in [110]. When adapted to our model, it first computes for each row i the value $MIMSR$ (maximum interference to minimum signal ratio), defined by

$$MIMSR(i) = \max\{\frac{\beta G(i,j)}{G(i,i)} | j \neq i \wedge j \; not \; scheduled\}$$

and removes links with MIMSR above a threshold ζ. We present here a simplified and centralized version, which produces schedules of at most the same length as the original algorithm.

$$CON : MIMSR(k) > \zeta.$$

LISRA (Limited Information Stepwise Removal Algorithm), described in [119], postpones the transmission of the links with the lowest $SINR$ when all sender transmit with equal power, to increase the probability for the remaining links to reach the $SINR$ threshold[3]. To generate schedules with LISRA we replace Step 5 of the generic with

5a: set $\mathbf{P} = 1$ and compute $SINR$;
5b: remove links γ_k for which $\min_i SINR(i) = SINR(k)$;

$$CON : SINR(k) \; is \; minimal \; for \; k.$$

These algorithms have all been tested in situations with nodes distributed uniformly at random. No worst case analysis has been done and the authors do not give any guarantees on their behavior. To prove our point we construct an example where the schedules these algorithms produce are extremely long.

[3]In its original version, step 3 contains the execution of an iterative distributed algorithm based on locally available information. The number of rounds is fixed beforehand, hence the quality of the results depend on the convergence speed of the algorithm. As we are most interested in the maximal length of the schedules LISRA produces, we replace the algorithm in step 3 by a (centralized) eigenvalue decomposition.

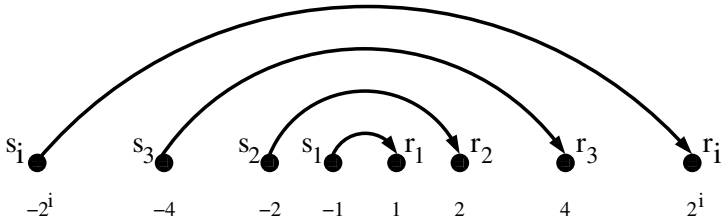


Figure 4.2: A set of communication request of exponentially increasing length. For this instance, all existing power control and scheduling algorithms assign a separate time slot to each link, even though the 3-disturbance is $\chi_3 = 1$ and no more than $O(\log n)$ time slots are necessary

Consider a scenario S with $k = \frac{n}{2}$ communication requests where all the sender and receiver nodes are situated on a straight line with the following distance to 0: Sender node $s_i = -2^i$, receiver node $r_i = 2^i, \forall 0 < i \leq k$. This situation is depicted in Figure reffig:worstcasescenario. We set $\alpha = 3$, the noise level $N = 0$ and the minimum $SINR$ necessary for successful transmission to $\beta = 2$. For this situation all the algorithms described above perform poorly, namely they schedule each link individually and require $\Omega(n)$ time slots, even though we prove $O(\log n)$ time slots to be sufficient. Because the 3-disturbance of the above scenario S is $\chi_3 = 1$, our example demonstrates that these algorithms exhibit severe worst-case problems even in networks with low disturbance.

Theorem 4.4. *SRA, LISRA, SMIRA and WCRP produce a schedule of length* $\Omega(n)$ *for the scenario* S *in which the* 3*-disturbance* χ_3 *is 1.*

Proof. Starting with SRA, we prove the claim for each algorithm individually.

SRA: As we cannot schedule all links in the same slot, we compute the column and row sums of Z to decide which links we postpone to subsequent time slots. The sum for row i is

$$R_i = \sum_{j=1}^{n} z(i,j) = \sum_{j=1}^{n} \left(\frac{2^{i+1}}{2^j + 2^i} \right)^{\alpha},$$

which is maximal when $i = n$. Analogously the sum for column i is

$$C_i = \sum_{j=1}^{n} z(j,i) = \sum_{j=1}^{n} \left(\frac{2^{j+1}}{2^j + 2^i} \right)^{\alpha}.$$

This sum reaches its maximum when $i = 1$, since i only appears in the denominator. Hence we have to determine $\max\{R_n, C_1\}$.

The summands of C_1 grow with j whereas the summands of R_n decrease. As a consequence we can simplify the analysis by comparing $\frac{2^{n+1}}{2^{n-j+1}+2^n}$ to $\frac{2^{j+1}}{2^j+2}$.

$$\frac{2^{n+1}}{2^{n-j+1}+2^n} = \frac{2^j}{1+2^{j-1}} = \frac{2^{j+1}}{2+2^j} \qquad \forall 0 < j \leq n.$$

Hence we know that the largest row sum is equal to the largest column row, which causes either the shortest or the longest link to be removed from the set of links to schedule in the next time slot. Without loss of generality we assume that we postpone the transmission of the shortest link.

Without the first link we have to deal with almost the same situation, the only difference is the fact that the sums start with $j = 2$ instead of 1. Again we remove the shortest link. This game continues until only one link is left, since two links next to each other cannot be scheduled in the same slot.

Lemma 4.5. *Two links l_i and l_{i+1} cannot be scheduled in the same slot.*

Proof. Let $l_i = (-2^i, 2^i), l_j = (-2^j, 2^j)$. We compute

$$Z = \begin{pmatrix} 1 & \left(\frac{2^{i+1}}{2^j+2^i}\right)^\alpha \\ \left(\frac{2^{j+1}}{2^j+2^i}\right)^\alpha & 1 \end{pmatrix}$$

and set $j = i + 1$. Now the larger eigenvalue is

$$\begin{aligned} \lambda^* &= 1/2\left(z_{1,1} + z_{2,2} + \sqrt{4z_{1,2}z_{2,1} + (z_{1,1} - z_{2,2})^2}\right) \\ &= 1/2\left(1 + 1 + \left(\sqrt{4 \cdot 2^{i+j+2}/(2^i + 2^j)^2}\right)^\alpha\right) \\ &\overset{j=i+1}{=} 1 + \left(\frac{\sqrt{2^{2i+3}}}{2^i + 2^{i+1}}\right)^\alpha = 1 + \left(\frac{\sqrt{8}}{3}\right)^\alpha > 1.83. \end{aligned}$$

Consequently $SINR^* = \frac{1}{\lambda^* - 1} < 1.19$, implying that the links l_i and l_{i+1} cannot be transmitted simultaneously. □

We can derive from the above, that SRA schedules all links individually, i.e. the length of the schedule is $\Omega(n)$.

SMIRA: The transmission of link l_i is postponed if either the interference received and the interference caused by link l_i is above a certain threshold. As the receiving node of link 1 suffers from the highest level of interference we remove it. This situation occurs again in the next time slot, hence each link is scheduled individually, leading to a complexity of $\Omega(n)$.

WCRP: We compute the MIMSR value for each link i.

$$MIMSR(i) \quad = \quad \max_j \frac{\beta \cdot G(i,j)}{L \cdot G(i,i)} = \beta \cdot \max_i \left(\frac{2^{i+1}}{2^i + 2^j} \right)^\alpha .$$

As $MIMSR(i)$ cannot exceed $\beta 2^\alpha$, we define $\zeta = 10$. Hence all links apart from the three shortest links are removed. Let us assume for simplicity that those can be scheduled in one slot. If we repeat this step, again the three shortest links remain and we can conclude that this method produces a schedule of length $\lceil n/3 \rceil \in \Omega(n)$

LISRA: The same holds for LISRA, although with a slightly different reasoning. LISRA iteratively removes the link which achieves the lowest $SINR$ with equal power distribution until β is reached. In our example, the link to be postponed will always be the longest link. As we have seen above, two neighboring links cannot be scheduled in the same time slot, hence LISRA also needs $\Omega(n)$ slots.

□

All four algorithms produce a schedule of length $\Omega(n)$ for this example. However, it is possible to construct a much shorter schedule. We present a schedule that needs as few as $O(\log n)$ time slots for the $n/2$ links.

Theorem 4.6. *There exists a scheduling and power assignment scheme which produces a schedule of length $O(\log n)$ for scenario S for all $n > 16$.*

Proof. Consider the schedule where every $\log n^{th}$ link starting with 1 is selected for transmission in slot 1, every $\log n^{th}$ link starting with 2 for slot 2, etc. More formally, we schedule $\{l_t, l_{t+\log n}, l_{t+2\log n}, \ldots\}$ in time slot t. We construct a power assignment $P(s_i)$ such that every link exceeds a signal-to-interference-ratio of 2.

Let us have a closer look at the set L_t containing the links scheduled for time slot t. There are at most $\lceil \frac{n}{2\log n} \rceil$ links scheduled in this slot, of which we select link $l_i = (-2^i, 2^i)$, the τ_i^{th} longest link. Consider the assignment $P(s_i) = (2n)^{\tau_i} 2^{\alpha(i+1)}$ to s_i and recall that $SINR(i) = P_r(s_i) / \sum_{l_j \in L_t \setminus \{l_i\}} I_i(s_j)$.

We note that the largest amount of interference is caused by the neighboring links $l_{(i-\log n)}$ and $l_{(i+\log n)}$. Moreover, the interference power is cut in half for each link further away from l_i.

Claim 4.7. *The following two inequalities hold for all values $0 < j < n$ and $n \geq 8$*

$$\begin{aligned} I_i(s_{i-j\log n}) &> 2I_i(s_{i-(j+1)\log n}) \\ I_i(s_{i+j\log n}) &> 2I_i(s_{i+(j+1)\log n}). \end{aligned}$$

Proof. The first inequality holds because of

$$\begin{aligned}
\frac{I_i(s_{i-j\log n})}{I_i(s_{i-(j+1)\log n})} &= \frac{P(s_{i-j\log n})g(s_{i-j\log n}, r_i)}{P(s_{i-(j+1)\log n})g(s_{i-(j+1)\log n}, r_i)} \\
&= \frac{(2n)^{\tau_i+j}2^{\alpha(i-j\log n+1)}(2^i+2^{i-j\log n})^{-\alpha}}{(2n)^{\tau_i+j+1}2^{\alpha(i-(j+1)\log n+1)}(2^i+2^{i-(j+1)\log n})^{-\alpha}} \\
&= \frac{n^\alpha}{2n}\frac{(1+n^{-j-1})^\alpha}{(1+n^{-j})^\alpha} \\
&> \frac{n^{\alpha-1}}{4} \geq 2 \qquad \forall n \geq 8.
\end{aligned}$$

The other inequality can be proved analogously.

$$\begin{aligned}
\frac{I_i(i+j\log n)}{I_i(i+(j+1)\log n)} &= \frac{P(i+j\log n)G(i, i+j\log n)}{P(i+(j+1)\log n)G(i, i+(j+1)\log n)} \\
&= \frac{(2n)^{\tau_i-j}2^{\alpha(i+j\log n+1)}(2^i+2^{i+j\log n})^{-\alpha}}{(2n)^{\tau_i-j-1}2^{\alpha(i+(j+1)\log n+1)}(2^i+2^{i+(j+1)\log n})^{-\alpha}} \\
&= \frac{2n}{n^\alpha}\frac{(1+n^{j+1})^\alpha}{(1+n^j)^\alpha} \\
&> \frac{2n}{n^\alpha}\frac{n^\alpha}{2^\alpha} \\
&= \frac{n}{2^{\alpha-1}} \geq 2 \qquad \forall n \geq 8.
\end{aligned}$$

□

Applying Claim 4.7 we can bound $SINR(i)$ as follows

$$\begin{aligned}
SINR(i) = \frac{P_r(s_i)}{\sum_{l_j \in L_t \setminus \{l_i\}} I_i(s_j)} &\geq \frac{P_r(s_i)}{2(I_i(s_{i-\log n})+I_i(s_{i+\log n}))} \\
&= \frac{\frac{(2n)^{\tau_i}2^{\alpha(i+1)}}{2^{\alpha(i+1)}}}{2(\frac{(2n)^{\tau_i+1}2^{\alpha(i-\log n+1)}}{(2^i+2^{i-\log n})^\alpha} + \frac{(2n)^{\tau_i-1}2^{\alpha(i+\log n+1)}}{(2^i+2^{i+\log n})^\alpha})} \\
&= \frac{(2n)^{\tau_i}}{2\frac{(2n)^{\tau_i-1}2^{\alpha(i-\log n+1)}}{2^{\alpha(i-\log n)}}(\frac{(2n)^2}{(2^{\log n}+1)^\alpha} + \frac{2^{2\alpha\log n}}{(2^{2\log n}+2^{\log n})^\alpha})} \\
&= \frac{2n}{2^{\alpha+1}(\frac{(2n)^2}{(n+1)^\alpha} + \frac{n^{2\alpha}}{(n^2+n)^\alpha})} \\
&= \frac{n(n+1)^\alpha}{2^\alpha(4n^2+n^\alpha)} \geq 2 \qquad \forall n > 16.
\end{aligned}$$

Since the above holds for all communication requests in all slots, we have proved that this schedule allows the successful transmission of all links in $O(\log n)$ time slots. □

4.4 Low-Disturbance Scheduling Algorithm

In this section, we propose a novel scheduling algorithm, called the *Low-Disturbance Scheduling Algorithm(LDS)*, which achieves provable performance guarantees even in worst-case networks. In particular, given a network and a set of communication requests, LDS computes a schedule whose length is within a polylogarithmic factor of the network's disturbance.

Description

The algorithm consists of three parts: a pre-processing step, the main scheduling-loop, and a test-subroutine that determines whether a link is to be scheduled in a given time slot.

The purpose of the pre-processing phase is to assign two values $\tau(i)$ and $\gamma(i)$ to every request l_i. The value $\gamma(i)$ is an integer values between 1 and $\lceil \log(3n\beta) + \rho \log \alpha \rceil$. The idea is that only requests with the same $\gamma(i)$ values are considered for scheduling in the same iteration of the main scheduling-loop (Lines 2 and 3 of the main scheduling-loop). The second assigned value, $\tau(i)$, further partitions the requests. In particular, it holds that the length of all requests that have the same $\gamma(i)$ and $\tau(i)$ differ by at most a factor two. On the other hand, we show in Lemma 4.11 that if two requests l_i and l_j satisfy $\tau(i) < \tau(j)$, then the length of l_i, d_i, is at least by a factor $\frac{1}{2}(3n\beta\rho^{\alpha})^{\tau(j)-\tau(i)}$ longer than d_j. Generally speaking, the assignment of $\tau(i)$ ensures that the smaller the value $\tau(i)$ assigned to a requests l_i, the longer the corresponding communication link, and vice versa.

In summary, the pre-processing phase partitions the set of requests in such a way that two requests l_i and l_j that are assigned the same $\gamma(i)$ have either almost equal length (if, $\tau(i) = \tau(j)$) or very different length. This partition will turn out to be crucial in the actual scheduling process, which takes part in the subsequent main scheduling-loop.

Each for-loop iteration of the main scheduling-loop schedules the set of requests having the same $\gamma(i)$ values, denoted by $\mathcal{F}_k$. As long as not all requests of $\mathcal{F}_k$ have been scheduled, the algorithm considers the remaining requests in $\mathcal{F}_k$ in decreasing order of their length d_i. Specifically, the algorithm checks for each request whether it can safely be scheduled alongside the longer links that have already been selected. If a request is chosen to be scheduled in time slot t, it is added to L_t, otherwise it remains in $\mathcal{F}_k$.

The decision whether a request l_i is selected for scheduling or not takes place in the **allowed**$(\mathbf{l_i}, \mathbf{L_t})$ subroutine. For each (longer) request $l_j \in L_t$ that has already been chosen to be scheduled in time slot t, the subroutine checks three conditions. Only if none of them is violated, l_i is added to L_t. Notice, however, that the selection-criteria are significantly more complex than the simple "reuse-distance" argument that has been used in previous work (e.g. [38]).

Algorithm 4.2 The LDS Protocol for requests L

Pre-processing phase:

$\tau\text{cur} := 1$; $\gamma\text{cur} := 1$; $\text{last} := d_1$;
Consider all requests $l_i \in L$ in decreasing order of d_i:
for each $l_i \in L$ **do**
  if $\text{last}/d_i \geq 2$ **then**
    if $\gamma\text{cur} < \lceil \log(3n\beta) + \rho \log \alpha \rceil$ **then**
      $\gamma\text{cur} := \gamma\text{cur} + 1$;
    else
      $\gamma\text{cur} := 1$; $\tau\text{cur} := \tau\text{cur} + 1$;
    fi
    $\text{last} := d_i$;
  fi
  $\gamma(i) := \gamma\text{cur}$; $\tau(i) := \tau\text{cur}$;
od

Main scheduling-loop:

Define constant ν such that $\nu := 4N$;
$t := 1$;
for $k = 1$ **to** $\lceil \log(3n\beta) + \rho \log \alpha \rceil$ **do**
  Let $\mathcal{F}_k$ be the set of all requests l_i with $\gamma(i) = k$.
  while not all requests in $\mathcal{F}_k$ have been scheduled **do**
    $L_t := \emptyset$;
    Consider all $l_i \in \mathcal{F}_k$ in decreasing order of d_i:
      if $\mathit{allowed}(l_i, L_t)$ **then**
        $L_t := L_t \cup \{l_i\}$; $\mathcal{F}_k := \mathcal{F}_k \setminus \{l_i\}$
      fi
    Schedule all $l_i \in E_t$ in time slot t, assigning s_i
    a transmission power of $P_i = \nu \cdot d_i^\alpha \cdot (3n\beta\rho^\alpha)^{\tau(i)}$;
    $t := t + 1$;
  od
od

allowed$(\mathbf{l_i}, \mathbf{L_t})$

Define constant μ such that $\mu := 4\sqrt[\alpha]{\frac{120\beta(\alpha-1)}{\alpha-2}}$;
for each $l_j \in L_t$ **do**
  $\delta_{ij} := \tau(i) - \tau(j)$;
  if $\tau(i) = \tau(j)$ and $\mu \cdot d_i > d(s_i, s_j)$
  or $\tau(i) > \tau(j)$ and $d_i \cdot (3n\beta\rho^\alpha)^{\frac{\delta_{ij}+1}{\alpha}} > d(s_i, r_j)$
  or $\tau(i) > \tau(j)$ and $d_j/\rho > d(s_j, r_i)$
  then return false
od
return true

In particular, the second criterion states that l_i is scheduled only if for all longer requests $l_j \in L_t$, it holds that

$$d_i \cdot (3n\beta\rho^\alpha)^{\frac{\tau(i)-\tau(j)+1}{\alpha}} > d(s_i, r_j)$$

if $\tau(i) > \tau(j)$. That is, the distance that must be maintained between the sender s_i of l_i and the receiver of r_j of some $l_j \in L_t$ depends on the relative values of $\tau(i)$ and $\tau(j)$ assigned in the pre-processing phase.

The definition of the three selection-criteria guarantees that all simultaneously transmitted requests in a single time slot are received successfully by the intended receivers. Additionally, the subsequent analysis section shows that all requests can be scheduled efficiently even in worst-case networks.

Analysis

In this section, we prove that the LDS algorithm is both correct (i.e., all requests scheduled during the algorithm's execution are received successfully at the intended receivers) and fast. Specifically, we prove that every set of requests can be scheduled efficiently even in worst-case networks provided that the ρ-disturbance of the requests is small. As we show in Section 4.3, this distinguishes the LDS algorithm from all existing algorithms, that may perform badly even if the disturbance is small.

We begin with two simple lemmas that bound the amount of interference created by simultaneously scheduled senders s_j at an intended received r_i.

Lemma 4.8. *Let l_i and l_j be two requests with $\tau(i) \neq \tau(j)$ the algorithm selects for the same time slot. The interference at r_i created by s_j is at most*

$$I_r(s_j) \leq \nu \cdot \rho^{\alpha\tau(i)} \cdot (3n\beta)^{\tau(i)-1},$$

where $\nu = 4N$.

Proof. We distinguish two cases:
a) $\tau(i) < \tau(j)$. In this case, we know that $d_i > d_j$ by the definition of Line 6 in the main scheduling-loop. Hence, by the time l_j is added to L_t by the **allowed**$(\ell_\mathbf{j}, \mathbf{L_t})$ subroutine, l_i is already in L_t. Because **allowed**$(\ell_\mathbf{j}, \mathbf{L_t})$ evaluated to **true**, the distance $d(s_j, r_i)$ is at least

$$d_j \cdot (3n\beta\rho^\alpha)^{\frac{\delta_{ij}+1}{\alpha}},$$

where $\delta_{ij} := \tau(j) - \tau(i)$. Hence the interference of s_j at r_i is at most

$$\begin{aligned} I_r(s_j) &= \frac{P_j}{d(s_j, r_i)^\alpha} \leq \frac{\nu \cdot d_j^\alpha \cdot (3n\beta\rho^\alpha)^{\tau(j)}}{d_j^\alpha \cdot (3n\beta\rho^\alpha)^{\delta_{ij}+1}} \\ &= \nu \cdot (3n\beta\rho^\alpha)^{\tau(i)-1}, \end{aligned}$$

which is smaller than the upper-bound claimed in the lemma.
b) $\tau(i) > \tau(j)$. In this case, it holds that $d_i < d_j$. Because both links have been selected by the algorithm, it follows that $d(s_j, r_i) \geq d_j/\rho$. Furthermore, it holds that $\tau(i) \geq \tau(j) + 1$, thus the maximum amount of interference that can be caused by s_j at r_i is

$$\begin{aligned} I_r(s_j) &= \frac{P_j}{d(s_j, r_i)^\alpha} \leq \frac{\nu \cdot d_j^\alpha \cdot (3n\beta\rho^\alpha)^{\tau(j)}}{(d_j/\rho)^\alpha} \\ &= \nu \cdot \rho^{\alpha(\tau(j)+1)} \cdot (3n\beta)^{\tau(j)} \\ &\leq \nu \cdot \rho^{\alpha\tau(i)} \cdot (3n\beta)^{\tau(i)-1}. \end{aligned}$$

□

The next lemma bounds the total interference created by all nodes transmitting simultaneously for which $\tau(i) = \tau(j)$.

Lemma 4.9. *Given a request l_i, the total interference I_r^0 at r_i created by all senders s_j transmitting simultaneously for which $\tau(i) = \tau(j)$ is at most $I_r^0 \leq \frac{\nu}{4}\beta^{\tau(i)-1}(3n\rho^\alpha)^{\tau(i)}$.*

Proof. By the pre-processing phase, it holds that if both $\tau(i) = \tau(j)$ and $\gamma(i) = \gamma(j)$, then $\frac{d_j}{2} \leq d_i \leq 2d_j$ is satisfied. Thus, all requests have roughly the same lengths and we can bound the total interference using a standard area argument. Specifically, by Line 3 of the **allowed**$(\mathbf{l_i}, \mathbf{L_t})$ subroutine, l_i and l_j being scheduled in the same time slot implies that $\mu \cdot d_i > d(s_i, s_j)$, where $\mu := 4\sqrt[\alpha]{120\beta(\alpha-1)/\alpha-2}$. Now, consider all concurrently transmitting nodes s_j for which $\tau(i) = \tau(j)$ and consider disks D_j of radius $\frac{\mu d_i}{4}$ centered at each such sender. Because of the required spatial reuse distance and the fact that the length of two requests differs by at most a factor two, it holds that $d(s_j, s_{j'}) > \frac{\mu d_i}{2}$ and hence, disks D_j do not overlap. The area of each such disk is $A(D_i) \geq (\frac{\mu d_i}{4})^2\pi$.

Consider rings R_k of width μd_i around r_i, consisting of all senders s_j transmitting simultaneously for which $\tau(i) = \tau(j)$ and

$$\frac{k\mu}{2}d_i \leq d(s_j, r_i) \leq \frac{(k+1)\mu}{2}d_i.$$

Observe that by the first condition of the subroutine, R_1 must be empty. Consider a ring R_k and the transmitters contained in it. All corresponding disks D_i must be entirely located in an "extended" ring R_k^* of area

$$\begin{aligned} A(R_k^*) &= \left[\left(\frac{(k+1)\mu d_i}{2} + \frac{\mu d_i}{2}\right)^2 - \left(\frac{k\mu d_i}{2} - \frac{\mu d_i}{2}\right)^2\right]\pi \\ &= \frac{3(2k+1)}{4}\mu^2 d_i^2 \pi. \end{aligned}$$

The distance of a sender s_j in R_k from r_i has a lower bound of $\frac{k\mu}{2}d_i$. Furthermore, each such sender transmits at a power at most $\nu \cdot (2d_i)^\alpha \cdot (3n\beta\rho^\alpha)^{\tau(i)}$. Using the fact that the disks D_i do not overlap, we can bound the interference at r_i from nodes in ring R_k by

$$\begin{aligned} I_r^0(R_k) &\leq \frac{A(R_k^*)}{A(D_i)} \cdot \frac{\nu(3\beta n\rho^\alpha)^{\tau(i)} \cdot (2d_i)^\alpha}{(\frac{k\mu}{2}d_i)^\alpha} \\ &< \frac{12(2k+1)\nu(3\beta n\rho^\alpha)^{\tau(i)} \cdot 2^{2\alpha}}{(k\mu)^\alpha} \\ &\leq \frac{30\nu(3\beta n\rho^\alpha)^{\tau(i)} \cdot 2^{2\alpha}}{k^{\alpha-1}\mu^\alpha}, \end{aligned}$$

where the last inequality follows because only rings where $k \geq 2$ need to be considered. Summing up the interference generated by all rings results in a total interference of

$$\begin{aligned} I_r^0 &< \sum_{k=1}^{\infty} I_r^0(R_k) \leq \frac{30\nu(3\beta n\rho^\alpha)^{\tau(i)} \cdot 2^{2\alpha}}{\mu^\alpha} \sum_{k=1}^{\infty} \frac{1}{k^{\alpha-1}} \\ &< \frac{30\nu(3\beta n\rho^\alpha)^{\tau(i)} \cdot 2^{2\alpha}}{\mu^\alpha} \cdot \frac{\alpha-1}{\alpha-2} \\ &< \frac{\nu}{4}\beta^{\tau(i)-1}(3n\rho^\alpha)^{\tau(i)}, \end{aligned}$$

where the second-to-last inequality follows from a bound on Riemann's zeta-function and the last one from plugging in the definition of μ. This concludes the proof. □

Using the previous two lemmas, it can now be shown that every message scheduled for transmission by the algorithm can be decoded successfully by the intended receiver.

Theorem 4.10. *The schedule computed by the algorithm allows all requests to be successfully received by the intended receiver.*

Proof. Using Lemmas 4.8 and 4.9, we bound the total interference I_r created by concurrent senders as

$$\begin{aligned} I_r &\leq \frac{\nu}{4}\beta^{\tau(i)-1}(3n\rho^\alpha)^{\tau(i)} + \sum_{s_j:\tau(i)\neq\tau(j)} \nu\rho^{\alpha\tau(i)}(3n\beta)^{\tau(i)-1} \\ &\leq (\nu/4 + \nu/3)\,(3n\rho^\alpha)^{\tau(i)}\beta^{\tau(i)-1}. \end{aligned}$$

The theorem follows from verifying that the resulting $SINR$ is sufficiently high and by noting that every request is scheduled for transmission exactly once by the algorithm.

$$SINR(r_i) \geq \frac{\nu \cdot (3n\beta\rho^\alpha)^{\tau(i)}}{N + \left(\frac{\nu}{3} + \frac{\nu}{4}\right)(3n\rho^\alpha)^{\tau(i)}\beta^{\tau(i)-1}} > \beta.$$

□

So far, we have proven that the produced schedule is correct in the sense that all messages are actually received successfully. It now remains to show that the schedule is short and includes all requests. For this reason, we bound the number of time slots required to schedule all requests that have the same $\gamma(i)$ value. That is, we bound the amount of time used for one iteration of the for-loop in the main scheduling-loop. We begin with two simple lemmas.

Lemma 4.11. *Consider two requests l_i and l_j with $\gamma(i) = \gamma(j)$. If $\tau(i) \geq \tau(j)$ it holds that*

$$d_j \;\geq\; 1/2(3n\beta\rho^{\alpha})^{\tau(i)-\tau(j)} \cdot d_i.$$

Proof. If two requests l_i and l_j have the same γ value but different τ values, $\gamma(i)$ has been increased at least $(\tau(i) - \tau(j))\lceil \log(3n\beta) + \rho \log \alpha \rceil$ times since processing l_j. The reason is that $\gamma(i)$ must be increased exactly $\lceil \log(3n\beta) + \rho \log \alpha \rceil$ times (and reset to 0 once) in order to reach $\gamma(i) = \gamma(j)$ for the next higher value of τ. Due to Line 4, each but one such increase implies a halving of the length d_j. Hence,

$$d_j \geq d_i \cdot 2^{(\tau(i)-\tau(j))(\log(3n\beta)+\rho \log \alpha)} \geq d_i \cdot (3n\beta\rho^{\alpha})^{\tau(i)-\tau(j)}.$$

□

Lemma 4.12. *In any disk D of radius R, there can be at most χ_ρ receivers r_i of requests l_i with length $d_i \geq 2\rho R$.*

Proof. If $d_i \geq 2\rho R$ for all l_i, the disk of radius d_i/ρ around each receiver fully covers D. The claim now follows from the definition of χ_ρ. □

In order to bound the number of time slots required to schedule all requests in the same iteration of the main loop, we define the notion of *blocking requests*.

Definition 4.13. *l_j is a* blocking request *for l_i if $\gamma(i) = \gamma(j)$, $d_j \geq d_i$, and* **allowed**$(\mathbf{l_i}, \mathbf{L_t})$ *evaluates to false if $l_j \in L_t$. B_i denotes the set of blocking requests of l_i.*

Consequently, blocking requests $l_j \in B_i$ are those requests that can "block" a request l_i from being scheduled in a given time slot. Because each such blocking request can prevent l_i from being scheduled only in a single time slot (when it is scheduled itself), it holds that l_i is scheduled in time slot $|B_i| + 1$ or earlier of the for-loop iteration when requests with $\gamma(i)$ are scheduled. We distinguish three kinds of blocking requests, depending on which of the three conditions in the **allowed**$(\mathbf{l_i}, \mathbf{L_t})$ subroutine is responsible for the blocking, and we bound the number of blocking requests in each category independently.

Lemma 4.14. *Let B_i^1 be the set of blocking requests $l_j \in B_i$ with $\tau(i) = \tau(j)$ and $\mu d_i > d(s_i, s_j)$. For all l_i it holds that $|B_i^1| \leq 4\rho^2(\mu+2)^2\chi_\rho$.*

Proof. From $\tau(i) = \tau(j)$, it follows by Lemma 4.11 that $d_i \leq d_j \leq 2d_i$ for all $l_j \in B_i^1$. By Lemma 4.12, we know that there can be at most χ_ρ receivers of blocking requests with length at least d_i in any disk of radius $d_i/(2\rho)$. Because $\mu d_i > d(s_i, s_j)$ holds for any blocking request in B_i^1, any receiver corresponding to a blocking request must be located inside a disk of radius $(\mu+2)d_i$ centered at s_i. Thus,

$$|B_i^1| \;\leq\; \chi_\rho \cdot \frac{\pi(\mu+2)^2 d_i^2}{\frac{1}{(2\rho)^2}\pi d_i^2} \;=\; 4\rho^2(\mu+2)^2\chi_\rho.$$

□

The next lemma is key to our worst-case result and bounds the number of blocking requests that prevent a shorter request by the second condition of the **allowed**$(\mathbf{l_i}, \mathbf{L_t})$ subroutine.

Lemma 4.15. *Let B_i^2 be the set of blocking requests $l_j \in B_i$ with $\tau(i) > \tau(j)$ and $d_i \cdot (3n\beta\rho^\alpha)^{\delta_{ij}+1/\alpha} > d(s_i, r_j)$. For all l_i it holds that*

$$|B_i^2| \leq 16\log(n+1)\chi_\rho.$$

Proof. First we show that for any integer $\varphi \geq -1$, there can be $O(\chi_\rho)$ different blocking requests $l_j \in B_i^2(\varphi)$ where

$$(3n\beta\rho^\alpha)^{\alpha^\varphi} \cdot d_i \;<\; d(s_i, r_j) \;\leq\; (3n\beta\rho^\alpha)^{\alpha^{\varphi+1}} \cdot d_i.$$

By the definition of the second condition in the **allowed**$(\mathbf{l_i}, \mathbf{L_t})$ subroutine, each such request $l_j \in B_i^2(\varphi)$ must satisfy $\frac{\delta_{ij}+1}{\alpha} > \alpha^\varphi$, and hence $\delta_{ij} \geq \alpha^{\varphi+1}$. By Lemma 4.11, we know that each such blocking request $l_j \in B_i^2(\varphi)$ with $d(s_i, r_j)$ in the range specified above must be of length at least $d_j \geq \frac{1}{2}(3n\beta\rho^\alpha)^{\alpha^{\varphi+1}} \cdot d_i$.

It remains to show that there can be at most $O(\chi_\rho)$ such requests $l_j \in B_i^2(\varphi)$. For simplicity, define $K := (3n\beta\rho^\alpha)^{\alpha^{\varphi+1}} \cdot d_i$. By Lemma 4.12 and the above lower bound on d_j, at most χ_ρ receivers of requests in $B_i^2(\varphi)$ can be in any disk of radius $\frac{K}{4\rho}$. By definition all these receivers r_j must be within distance K of s_i, thus that there can be at most $16\rho^2\chi_\rho$ blocking requests in $B_i^2(\varphi)$ by the classic area argument.

We know that for any integer $\varphi > -1$, there are at most $16\rho^2\chi_\rho$ blocking requests in $B_i^2(\varphi)$. The value δ_{ij} between two requests l_i and l_j cannot exceed n and hence, the furthest distance $d(s_i, r_j)$ of any blocking request l_j can be $(3n\beta\rho^\alpha)^{\frac{n+1}{\alpha}} d_i$. It follows that $|B_i^2(\varphi)| = 0$ for all $\varphi > \frac{n+1}{\alpha}$. Finally, because $\alpha^{(\varphi+1)} > \frac{n+1}{\alpha}$ for some $\varphi \geq \log_\alpha(n+1)$, it follows that there are at most

$O(\log n)$ many "rings", each of which can contain at most $16\rho^2\chi_\rho$ blocking receivers. Hence,

$$|B_i^2| \;=\; \sum_{\varphi:=-1}^{\infty} |B_i^2(\varphi)| \;\leq\; \log_\alpha(n+1)\cdot 16\rho^2\chi_\rho.$$

□

Finally, we bound the number of blocking requests that can block a request r_i due to the third constraint in the **allowed**$(\mathbf{l_i}, \mathbf{L_t})$ subroutine.

Lemma 4.16. *Let B_i^3 be the set of requests $l_j \in B_i$ with $\tau(i) > \tau(j)$ and $\frac{d_j}{\rho} > d(s_j, r_i)$. It holds $|B_i^3| \leq 6\chi_\rho$ $\forall l_i$.*

Proof. Assume for contradiction that there are more than $6\chi_\rho$ such blocking requests $l_j \in B_i^3$. For each of these $d_j > d_i$. Partition the area around r_i into cones of angle $\pi/3$. At least one of these cones must contain the senders s_j of $\chi_\rho + 1$ or more blocking requests. The angle of this cone being $\pi/3$, the distance of the furthest such sender s_j' to each of the other blocking senders s_j in this cone is at most $d(s_j', s_j) < d(s_j', r_i)$, and hence, $d(s_j', s_j) < d_i/\rho < d_j'/\rho$. There are at least $\chi_\rho + 1$ senders within distance d_j'/ρ of s_j', which contradicts χ_ρ's definition. □

As every blocking request can block a request l_i at most once, we combine the above and prove the following theorem.

Theorem 4.17. *The number of time slots required by Algorithm 4.2 to successfully schedule all requests $l_i \in L$ is at most $O\left(\chi_\rho\rho^2 \log n \cdot (\log n + \rho)\right)$.*

Proof. By Lemmas 4.14, 4.15, and 4.16, any request l_i can be blocked by at most

$$B_i^1 + B_i^2 + B_i^3 \leq 4\rho^2(\mu+2)^2\chi_\rho + 16\rho^2\log(n+1)\chi_\rho + 6\chi_\rho$$

blocking requests. Thus, after at most $O(\chi_\rho\rho^2 \cdot \log n)$ iterations of the while-loop, all requests having the same $\gamma(i)$ value are scheduled successfully. The theorem follows as the number of for-loop iterations is $\lceil \log(3n\beta) + \rho\log\alpha \rceil$. □

Let us now examine the schedule our LDS-algorithm creates for the instance used to illustrate the inefficiency of previous algorithms. The 3-disturbance χ_3 of setting S is 1. Consequently, we obtain a schedule of length $O(\log^2 n)$ by plugging in the value $\rho = 3$ into the bound of Theorem 4.17. Notice that this is *exponentially shorter* than the schedules generated by any uniform or linear power assignment algorithm as well as any of the known link removal heuristics.

Corollary 4.18. *For $\rho = 3$, the LDS scheduling algorithm produces a schedule of length $O(\log^2 n)$ for scenario S.*

The LDS algorithm thus significantly outperforms existing scheduling strategies in worst-case scenarios. Nonetheless, the analysis of the power assignment $P(\cdot)$ of Theorem 4.6 demonstrates that an even better solution with complexity $O(\log n)$ exists. Hence, the aim for future research remains to devise algorithms, with results even closer to the optimum.

4.5 Concluding Remarks

In this chapter, we have shown that all scheduling algorithms studied so far may have an extremely suboptimal performance in worst-case networks. In order to ameliorate this situation, we propose the LDS scheduling algorithm. By employing a novel power assignment scheme and reuse distance criterion, our algorithm achieves a provably efficient performance in any network and request setting that features low disturbance.

In general, it can be argued that the network topologies and request sequences found in real-world applications may not have an explicit worst-case structure. We hope, however, that our power assignment strategy in combination with the theoretical insights gained from our worst-case analysis will ultimately lead to an increase in bandwidth and capacity beyond heuristics in real networks. Further investigation in this direction are bound to prove useful in areas such as wireless mesh networks, sensor networks, or even cellular networks.

5

Delay-Sensitive Aggregation

This chapter studies the fundamental trade-off between delay and communication cost in networks. We consider an online optimization problem where nodes are organized in a tree topology. The nodes seek to minimize the time until the root is informed about the changes of their states and to use as few transmissions as possible at the same time. We derive an upper bound on the competitive ratio of $O(\min(h, c))$ where h is the tree's height, and c is the transmission cost per edge. Moreover, we prove that this upper bound is tight in the sense that any oblivious algorithm has a ratio of at least $\Omega(\min(h, c))$. For chain networks, we prove a tight competitive ratio of $\Theta(\min(\sqrt{h}, c))$. Furthermore, we introduce a model for value-sensitive aggregation, where the cost depends on the number of transmissions and the error at the root.

5.1 Time and Energy Trade-Off

The analysis of distributed algorithms often revolves around time and message complexity. On the one hand, we want our distributed algorithms to be fast, on the other hand, communication should be minimized. Problems often ask to optimize one of the two—and treat the other only as a secondary target. However, there are situations where time and message complexity are equally important.

In this chapter, we study such a case known as *distributed aggregation*. Nodes of a large distributed network may sense potentially interesting data which they are to report to a central authority (sink). Not only should the data make its way fast through the network such that information is not unnecessarily delayed; but also, since message transmission is costly, one may reduce the number of transmissions by aggregating messages along the way. In other words, nodes may wait for further packets before forwarding them in order to decrease the number of transmission at the expense of a later arrival

of the information at the sink. This problem has many applications. In the past it was mostly studied in contexts such as control message aggregation. In the heyday of wireless networking the first application that comes to mind is perhaps sensor networking. Due to energy constraints, it is necessary to minimize the number of transmissions. At the same time, it is desirable to aim at minimizing the time until the nodes are informed about changes of measured values.

We assume that the communication network of the nodes forms a precomputed directed spanning tree on which information about events is passed to the root node (the sink). Data arrives at the nodes in an online (worst-case) fashion. The main challenge is to decide at what points in time the data should be forwarded to a parent in the tree.

Results

We prove that a simple algorithm achieves a competitive ratio of $O(\min(h, c))$ where h is the tree's height, and c is the transmission cost per edge. This improves an existing upper bound of $O(h \log(cn))$, where n is the network size. The examined algorithm is oblivious, i.e., decisions at each node are based solely upon the static local information available at the node. Being oblivious is a desirable property of distributed algorithms, since non-oblivious algorithms need dynamic updating mechanisms—a costly operation.

We also demonstrate that this upper bound is tight in the sense that there exist problem instances where any oblivious algorithm has a ratio of at least $\Omega(\min(h, c))$. Earlier work proved a lower bound of $\Omega(\sqrt{h})$ on a chain network. Therefore, we examine this topology more closely and show that chain networks are inherently simpler than general trees by giving a competitive ratio of $\Theta(\min(\sqrt{h}, c))$ for oblivious algorithms.

In the last part of this chapter, we present an event aggregation model which takes into account that nodes often have non-binary data to aggregate and greater differences between values need to be reported to the root faster than small differences. We present a model comprising this additional constraint as well as an oblivious algorithm achieving a competitive ratio of $\Theta(c/\epsilon)$ on a one-link network, where ϵ is the minimum difference between two values. We also describe an optimal offline algorithm (requiring polynomial time) for this model.

Related Work

The trade-off between delay and communication cost appears in various contexts, and plays a role in the design of algorithms for wireless sensor networks, for Internet transfer protocols, and also appears in organization theory. This section gives a brief overview of related work on this topic.

A basic problem in the design of Internet transfer protocols such as the TCP protocol concerns the acknowledgments (ACKs) which have to be sent by a receiver in order to inform the sender about the successful reception of packets: In many protocols, a delay algorithm is employed to acknowledge multiple ACK packets with a single message or to piggy-back the ACK on outgoing data segments [107]. The main objective of these algorithms is to save bandwidth (and other overhead at the sender and receiver) while still guaranteeing small delays. The problem of aggregating ACKs in this manner is also known as the *TCP acknowledgment problem* [36]. Karlin et al. [61] pointed out interesting relationships of the single-link acknowledgment problem to other problems such as *ski-rental*, and gave an optimal, $e/(e-1)$-competitive randomized online algorithm.

There are many variations of the theme, e.g., Albers et al. [3] seek to minimize the number of acknowledgments sent plus the *maximum* delay incurred for any of these packets. They propose a $\pi^2/6$-competitive deterministic algorithm for the single link, which is also a lower bound for any deterministic online algorithm. Frederiksen et al. [45] consider deterministic and randomized algorithms for bundling packets in a single-link network; their objective function measures the total time elapsed while packets are waiting at the leaf node, but have not been delivered yet.

There is also much literature on aggregation in sensor networks [67, 104, 106, 109, 116, 117]. E.g., Becchetti et al. [16] studied online and offline algorithms for scenarios where fixed deadlines must be met. They show that the offline version of the problem is strongly NP-hard and provide a 2-approximation algorithm. More complexity results for settings with fixed deadlines have been derived in [88].

Korteweg et al. [65] address a similar problem following a *bicriterion approach* which considers time and communication as two independent optimization problems: a (B, A)-bicriterion problem minimizes objective A under a budget on objective B. Inter alia, the authors prove that if r is the ratio between the maximum and the minimum delay allowed, then the competitive ratio of their algorithm is $(2h^{\lambda}, 2h^{1-\lambda} \log r)$ for any λ in $(0, 1]$.

The paper closest to our work is by Khanna et al. [63]. Our model is derived from [63] which investigates the task of centralized and decentralized online control message aggregation on weighted tree topologies. In particular, [63] presents a $O(h \log \alpha)$-competitive distributed algorithm, where h is the tree's height, and α is the sum of the weights of the tree's edges. Moreover, the authors show that any oblivious distributed online algorithm has a competitive ratio of at least $\Omega(\sqrt{h})$. In this chapter, we study the same algorithm and we give a new analysis for scenarios where the communication cost is c on all links, resulting in a better upper bound of $O(\min(h, c))$. We also derive a new generalized lower bound for edge cost which is different from h, and show that for any oblivious aggregation algorithm, the competitive ra-

tio is at least $\Omega(\min(h, c))$. Moreover, by taking into account many intrinsic properties of our algorithm, we show that for chain graphs an upper bound of $O(\min(\sqrt{h}, c))$ holds. This is asymptotically tight. Brito et al. [24] extended the work of Khanna et al. by proving general upper and lower bounds for the asynchronous and the centralized ACK aggregation problem. Interestingly, the asynchronous model yields higher lower bounds in the distributed case than the slotted model, e.g., the lower bound for the chain network is $\Omega(h)$ in the distributed asynchronous model, a factor of $\sqrt{h}$ greater than the bounds in the synchronous model.

Finally, our value-sensitive aggregation model is reminiscent of the recent "online tracking" work by Yi and Zhang [115]. The authors study a 1-lookahead scenario where Alice outputs a function $f : \mathbb{Z}^+ \to \mathbb{Z}^d$, and Bob, knowing all values at $t \leq t_{\text{now}}$ needs to guess $f(t)$ in an online fashion. Different scenarios are examined, and competitive strategies are presented that result in small errors only and save on communication cost. Our model differs from [115] as it describes a 0-lookahead model and as in their model, nodes are *forced* to send updates if the measured value differs sufficiently from the value stored at the root.

5.2 Model and Definitions

The network to be considered is modeled by a rooted tree $T = (V, E)$ of height h with root $r \in V$ and $n = |V|$ nodes. Every node u except for the root r (the *sink*) has a parent node v, i.e., an edge $(u, v) \in E$. The cost of transmitting a message over an edge is $c \geq 1$.

We assume that *events* occur at the leaf nodes $L \subset V$ (e.g., a control message arriving at a node, or a sensor node detecting an environmental change). We will refer to the information about the occurrence of a single event as an *event packet*. Leaf l creates an event packet p for every event that happens at l.

Eventually, all event packets have to be forwarded to the root. Instead of sending each packet $p \in \mathcal{P}$ individually to a node's parent after the event took place, nodes can retain packets and send a *message* m consisting of one or more packets together later, thus saving on communication cost as we have to pay for a link only once *per message* (rather than per event). Messages can be *merged* iteratively with other messages on their way to the root.

We consider a synchronous model where time is divided into time slots. In each slot, an arbitrary number of events can arrive at each node. For an event packet p, $t_l(p)$ denotes the time slot its corresponding event occurred at a node and $t_r(p)$ the time when it reaches the root. For each time slot an event waits at a node, we add one unit to the delay cost, i.e., the delay cost $dc(p)$ the event accumulates until reaching the root is $dc(p) = t_r(p) - t_l(p)$. Each message can only be forwarded one hop per time slot, i.e., a message

always has to wait one time slot at a node before being transmitted to the next node. Thus, the delay accumulated by an event is at least h_l, where h_l denotes the length of the path from the respective leaf l to the root. The total delay cost of all events accumulated up to time slot T is hence

$$dc_T = \sum_{p \in \mathcal{P}, t_r(p) \leq T} dc\,(p) + \sum_{p \in \mathcal{P}, t_r(p) > T} (T - t_l\,(p)).$$

Nodes can aggregate as many event packets as needed. At each time step t, a node may aggregate awaiting event packets and forward the resulting message to its parent. The cost of sending a message is c per edge no matter how many event packets it contains. Consequently, the total communication cost is the sum of the edge cost of all message transmissions. More formally, let S_t be the set of nodes sending out a message in time slot t, then the total communication cost cc_T up to time slot T is $cc_T = \sum_{t=1}^{T} |S_t|$. The total cost up to time T is the sum of both the delay and the communication cost,

$$cost_T = dc_T + cc_T.$$

Observe that the edge cost c allows us to weight delay and communication costs: a larger c implies that communication cost become relatively more important compared to the delay cost. Note that we neglect the energy consumption in idle listening mode and consider the nodes' transmission cost only. We believe that this is justified for networks where listening nodes have their radios turned off most of the time and only check for data transfers at the very beginning of each time slot.

Nodes do not know the future arrival time of events, and hence have to make the decisions on when to send messages *online*. We are in the realm of *competitive analysis* [22] and define the (strict) *competitive ratio* ρ achieved by an online algorithm $\mathcal{AGG}$ as the delay and communication cost of $\mathcal{AGG}$ divided by the total cost of an optimal offline algorithm $\mathcal{OPT}$.

Definition 5.1 (ρ-competitiveness)**.** *An online algorithm $\mathcal{AGG}$ is (strictly)* ρ-competitive *compared to an optimal offline algorithm $\mathcal{OPT}$ if for all input sequences S, i.e., all possible event arrival sequences,*

$$cost^{\mathcal{ALG}}(S) \leq \rho \cdot cost^{\mathcal{OPT}}(S).$$

The goal of an online algorithm designer is hence to devise algorithms minimizing ρ, as a small ρ describes the guaranteed worst-case quality of an algorithm.

We focus on *oblivious* online algorithms.

Definition 5.2 (Oblivious Algorithms)**.** *A distributed online algorithm $\mathcal{ALG}$ is called* oblivious *if the decisions by each node $v \in V$ whether to transmit a message solely depends on the number of events currently stored at node v and on the arrival times of the corresponding messages at node v.*

In particular, Definition 5.2 implies that the decisions of a node v do not depend on packets forwarded by v earlier or on v's location in the aggregation network.

5.3 Oblivious Online Algorithm

This section describes and analyzes the deterministic online algorithm $\mathcal{AGG}$ presented in [36, 63]. The algorithm is *oblivious* in the sense that decision are reached independently of the distance to the root (cf Definition 5.2). Essentially, the event aggregation algorithm $\mathcal{AGG}$ seeks to balance the total delay cost and the total communication cost. In order to do so, it aggregates information about multiple events into one message until the forwarding condition is satisfied. Whenever a new event occurs or a message arrives, it is merged with the message waiting for transmission at the node.

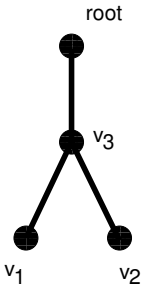
For a message m at node v, we define $delay_v(m,t)$, denoting the delay associated with message m currently stored at node v at time t. Informally, it is the sum of the accumulated delay cost of all the event packets the message m contains. In every time step a message remains at a node, $delay_v(m,t)$ is increased by the number of packets in the message. If a message arrives at a node where another message is already waiting to be forwarded, the two messages are merged. More formally, let a message m be a set of merged messages $\{m_1, \ldots, m_k\}$, where message m_i consists of $|m_i|$ packets and arrived at the current node in time slot t_i. The delay of message m at node v at time t is defined by

$$delay_v(m,t) := \sum_{i=1}^{k} |m_i|(t - t_i).$$

When executing algorithm $\mathcal{AGG}$, a node v forwards a message m to its parent as soon as the current accumulated delay exceeds the transmission cost:

$$delay_v\,(m,t) \geq c.$$

We demonstrate the execution of $\mathcal{AGG}$ on a simple example. Consider the tree and the event arrival sequence in Figure 5.1. There are two events occurring at leaf node v_1, one in time slot $t = 1$, one at time $t = 2$. Node v_2 receives two packets at $t = 2$. The transmission cost is set to $c = 3$. For this input sequence, node v_1 sends its two packets after time $t = 2$ and node v_2 after time $t = 3$, i.e., as soon as the accumulated delay reaches or exceeds $c = 3$. Node v_3 incurs a delay of two after the message from v_1 arrives. In the next time slot, v_3's delay cost increases to 6, as the message from v_2 arrives with two additional packets, so there are four messages at v_3 now.



events at node	v_1	v_2
$t = 1$	1	0
$t = 2$	1	2

delay at node	v_1	v_2	v_3
$t = 1$	1	0	0
$t = 2$	3	2	0
$t = 3$	0	4	2
$t = 4$	0	0	7
$t = 5$	0	0	0

Figure 5.1: Example execution of $\mathcal{AGG}$ where the transmission cost c is 3.

5.4 Tight Bound for Trees

Upper Bound

We establish a new upper bound of $O(\min(h, c))$ on the competitive ratio of $\mathcal{AGG}$ by a surprisingly simple analysis. Instead of calculating the delay and the communication cost the event packets accumulate, we focus on the messages $\mathcal{AGG}$ and $\mathcal{OPT}$ send. We proceed as follows. First, we investigate the competitiveness of $\mathcal{AGG}$ for a single link network, then tackle the chain network, and finally generalize our analysis to tree topologies.

Theorem 5.3. *On arbitrary trees, the competitive ratio of $\mathcal{AGG}$ is at most*

$$\rho = \frac{cost^{\mathcal{AGG}}}{cost^{\mathcal{OPT}}} \in O(\min(h, c)).$$

Proof. As a warm up, we consider the competitive ratio on a single link.

Lemma 5.4. *[36] The competitive ratio of $\mathcal{AGG}$ on a single link is at most* 2.

Proof. Consider a single link of cost c. Let t_i be the time slot $\mathcal{AGG}$ sends the message m_i containing information on k_i events. This transmission entails a communication cost of $cc_i^{\mathcal{AGG}} = c$. The total delay cost of the events in message m_i is $dc_i^{\mathcal{AGG}} = c + x_i$ for some $x_i \geq 0$, because m_i will be sent as soon as its events' delay exceeds c, and because there can be $(x_i + 1) \geq 0$ simultaneous event arrivals in the time slot immediately preceding t_i. Any optimal offline algorithm $\mathcal{OPT}$ has at least delay cost x_i as well. In addition, $\mathcal{OPT}$ incurs a cost of at least c for the event packets contained in m_i, either because of a transmission or due to the accumulated delay, thus $cost_i^{\mathcal{AGG}}/cost_i^{\mathcal{OPT}} \leq (2c + x_i)/(c + x_i)$. This expression is maximized for $x_i = 0$, implying a competitive ratio of 2. □

In a next step we analyze the chain network. Given a sequence of event arrivals at the leaf, we can compute when $\mathcal{AGG}$ sends messages to the leaf's parent node. We study the cost accumulated by these messages in groups of three messages. Then our results are generalized to trees.

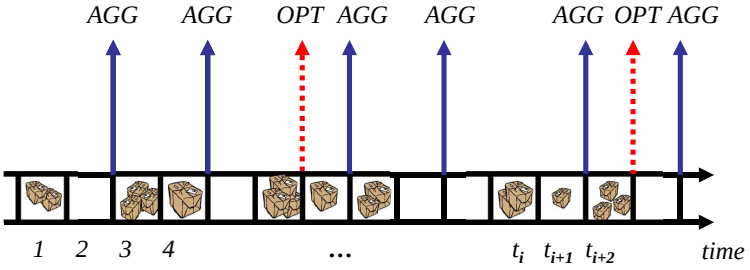


Figure 5.2: Illustration of a sequence of event arrivals at a leaf and the corresponding forwarding actions of the online algorithm $\mathcal{AGG}$ for $c = 4$. In addition, the time slots where the optimal offline algorithm forwards its messages are marked as well. Note that we cannot verify in this limited view of a leaf node how much better $\mathcal{OPT}$'s decisions are because of our lack of knowledge on the situation at the other nodes.

Lemma 5.5. *For the events occurring at a given leaf l, the competitive ratio is at most $O(\min(h_l, c))$.*

Proof. Given a sequence of event arrivals at the leaf, we can compute when $\mathcal{AGG}$ sends messages to the leaf's parent node. We define m_i^A to be the i^{th} message that leaf node l located at depth h_l sends to its parent. Let t_0 be the first time slot, and the time slot when message m_i^A is forwarded towards the root by l is denoted by t_i. Let the total number of messages that leaf l transmits be M_l^A. If such a message contains less than c packets it incurs a transmission cost of c and a delay cost of less than $2c$ per hop towards the root. If a message consists of more than c packets, the delay cost per hop is bounded by the number of packets. Thus, the total cost for $\mathcal{AGG}$ amounts at most $h_l(c + \max(2c, |m_i^A|))$ for message m_i^A.

In order to determine the minimum cost an offline algorithm accumulates, we divide the time into intervals of three message transmissions. More precisely, we consider time periods $[t_j, t_{j+3}]$, where $j \mod 3 = 0$. There are two possibilities for algorithm $\mathcal{OPT}$: Either it sends one or more messages in the period $[t_j, t_{j+3}]$ as well, yielding at least a transmission cost of c, or the delay accumulated by these packets is at least c. Either way, the cost accumulated by $\mathcal{OPT}$ at leaf l for these packets is at least c. In addition,

the delay cost accumulating until the packets arrive at the root is at least $(h_l - 1)(|m_j^A| + |m_{j+1}^A| + |m_{j+2}^A|)$, since every packet in m_j^A, m_{j+1}^A and m_{j+2}^A incurs a delay cost of at least one for each hop towards the root. Thus, $\mathcal{OPT}$'s total cost for a period $[t_j, t_{j+3}]$ is at least

$$c + (h_l - 1)(|m_j^A| + |m_{j+1}^A| + |m_{j+2}^A|).$$

Unless $M_l^A \mod 3 = 0$, we cannot consider all messages in groups of three. If there are one or two last messages that do not belong to such a group, the respective cost of $\mathcal{OPT}$ for the period $[t_{3\lfloor M_l^A/3 \rfloor}, t_{M_l^A}]$ is at least $c + (h_l - 1)|m_{M_l^A}|$ or $c + (h_l - 1)(|m_{M_l^A}| + |m_{M_l^A - 1}|)$.

This implies that for any given sequence of event arrivals at a leaf the competitive ratio is

$$\begin{aligned}
\rho &= \frac{cost^{\mathcal{AGG}}}{cost^{\mathcal{OPT}}} \\
&\leq \frac{\sum_{j=1}^{M_l^A} h_l(c + \max(2c, |m_j^A|))}{c\lceil M_l^A/3 \rceil + \sum_{j=1}^{M_l^A} h_l |m_j^A|} \\
&\leq \frac{3M_l^A h_l c + \sum_{j=1}^{M_l^A} h_l |m_j^A|}{cM_l^A/3 + \sum_{j=1}^{M_l^A} h_l |m_j^A|} \\
&< \frac{3M_l^A h_l c}{M_l^A(c/3 + h_l)} + 1 \\
&\in O\left(\min\left(h_l, c\right)\right).
\end{aligned}$$

□

Observe that we make no assumptions on the behavior of the optimal algorithm in Lemma 5.5. Moreover, we charge the optimal algorithm $\mathcal{OPT}$ only for the transmissions over the edges between the leafs and their neighbors when bounding $\mathcal{OPT}$'s communication cost, i.e., we basically exempt the optimal algorithm from paying for any other transmission. Furthermore, we ignore the cost $\mathcal{AGG}$ could potentially save by merging its messages on their way to the root: We assign the highest possible delay and communication cost to every message of $\mathcal{AGG}$. These properties allow us to extend Lemma 5.5 to trees without modification: more precisely, if we consider the longest path in the tree the statement holds for general trees as well. □

Lower Bound

We conclude our investigations of the tree network with a lower bound stating that $\mathcal{AGG}$ is asymptotically optimal compared to any *oblivious algorithm* (Definition 5.2). Recall that for oblivious algorithms, it holds that the wait

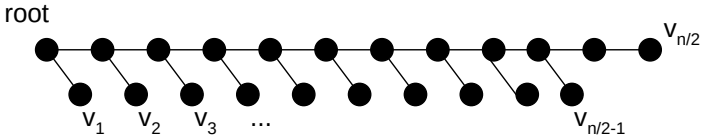


Figure 5.3: Lower bound topology.

time w only depends on the packet arrival time of the packets currently stored by a given node.

Theorem 5.6. *On the tree, any oblivious deterministic online algorithm has a competitive ratio of at least*

$$\rho = \frac{cost^{ALG}}{cost^{\mathcal{OPT}}} \in \Omega(\min(h, c)).$$

Proof. Consider the tree topology depicted in Figure 5.3 which consists of a chain network of $n/2 + 1$ nodes, where all nodes except for the two last ones have an additional neighbor (n even).The leaf nodes are referred to by $v_1, ..., v_{n/2}$.

Assume an input sequence where all leafs simultaneously get one packet. We now compute the minimum delay and communication cost any oblivious online algorithm ALG will accrue for this instance. Since ALG is oblivious, according to Definition 5.2, each leaf node v_i will send the packet to its parent after waiting for w time slots, where the packet arrives at time $w+1$. (The value of $w \geq 0$ depends on the chosen algorithm; observe that w is the same for all nodes, because we assume ALG to be oblivious.) From there, the packets leave at time $2w + 2$. This process is repeated until all packets reach the root. Generally, the packet of leaf node v_i will arrive at a node at distance j from v_i at time $j + jw$, and will stay there for w time slots. Observe that the packets of two nodes v_{i-1} and v_i are never merged into one message. Thus, ALG has communication cost in the order of $\Theta\left(h^2 c\right)$ and delay cost in the order of $\Theta\left(h^2\right)$.

The optimal algorithm $\mathcal{OPT}$ has to send at least one message over each edge, resulting in a communication cost in $\Theta\left(ch\right)$. The minimum delay cost until all packets reach the root is $\Theta\left(h^2\right)$. If all the packets are merged into one message on their way to the root, the delay and communication cost is in this order of magnitude. This yields the following lower bound on the competitive ratio

$$\rho = \frac{cost^{ALG}}{cost^{\mathcal{OPT}}} \in \Omega\left(\frac{h^2 + h^2 c}{2hc + h^2}\right),$$

which completes the proof. □

This result implies that $\mathcal{AGG}$ is asymptotically optimal in the sense that there exists no oblivious online algorithm achieving a better performance.

Discussion

Note that the above analysis for upper and lower bounds carries over to the model where events can appear at *any* node, not only at the leafs: The case where events occur at inner nodes can be reduced to a problem instance where events only occur at leafs by simply creating a new tree with additional leafs attached to any node where events occur; we have to map the arrival time slots to earlier time slots, ensuring that they arrive at the inner nodes at the correct time.

Theorem 5.3 can be compared to the results obtained in [63]. There, an upper bound of $O(h \log \alpha)$ is derived for the competitive ratio of $\mathcal{AGG}$, where α is the total communication cost (sum of edge weights) of the tree. If all edges have a weight c, this translates into $O(h \log(nc))$, which is $O\left(h^2 \log c\right)$ in balanced binary trees. In this case, the analysis presented above is better by a factor of $\Theta(h \log c)$ if $h < c$. In other networks, for instance, on chain topologies, the gap between the two bounds narrows, although there always remains a factor in the order of $\Omega(\log(\max(h, c))$: Let us only take the edges on the longest path into account for the upper bound of [63]. Thus the bound reduces to $O(h \log(hc))$. If $c > h$ this is in $O(h \log c)$ whereas the bound presented here is $O(h)$. If $c < h$ the bound from [63] gives $O(h \log h)$ and our analysis results in $O(c)$.

5.5 Tight Bound for Chains

Upper Bound

In order to obtain our upper bound for trees, the previous section has already briefly studied the competitiveness of $\mathcal{AGG}$ on chain networks. In the following, we show by a more detailed analysis taking into account many intrinsic properties of $\mathcal{AGG}$, that the bound can be improved. In particular, we leverage the fact that the algorithm can merge messages and forward them together, i.e., save on transmission cost. Moreover, a merged message moves faster towards the root since it now contains more packets and reaches the forwarding threshold in fewer time slots than the two separate messages consisting of fewer packets. Concretely, we prove that on the chain topology, $\mathcal{AGG}$ is $O(\sqrt{h})$-competitive, and that no oblivious algorithm can be less than $\Omega(\min(c, \sqrt{h}))$-competitive.

Theorem 5.7. *In chain graphs, the competitive ratio of $\mathcal{AGG}$ is*

$$\rho = \frac{cost^{\mathcal{AGG}}}{cost^{\mathcal{OPT}}} \in O\left(\min\left(\sqrt{h}, c\right)\right).$$

Proof. We use the following lemmata repeatedly.

Lemma 5.8. *When executing $\mathcal{AGG}$, a message consisting of n_i packets that arrive at the leaf at the same time reaches the node in distance l from the leaf after $l\lceil c/n_i \rceil$ time slots if the message does not merge with any other message.*

Proof. Let t_j denote the time the message spends at depth h_j, i.e., at the j^{th} node of its path towards the root. Since $\mathcal{AGG}$ is oblivious and no messages are merged, $t_j = \lceil c/n_i \rceil$ is the same for every node of the chain. Thus the number of time slots the message spends on a path of length l is $\sum_{j=0}^{l-1} t_j = l\lceil c/n_i \rceil$. □

Lemma 5.9. *If $\mathcal{AGG}$ does not merge any messages then there are at least*

$$\max\left(1, h\left\lceil \frac{c}{n_i} \right\rceil - (h-1)\left\lceil \frac{c}{n_{i+1}} \right\rceil\right)$$

time slots between the arrival of the packets of m_i^A and the arrival of the packets of m_{i+1}^A at the leaf node, if m_{i+1}^A contains more messages than m_i^A.

Proof. To ensure that m_i^A reaches the root separately even though it is followed by a larger message, m_i^A has to have arrived at the root before m_{i+1}^A arrives at the node in depth 1. By Lemma 5.8 we know that m_i^A reaches the root $h\lceil c/n_i \rceil$ time slots after its arrival at the leaf. If m_{i+1}^A arrives at the leaf δ_i time slots after the departure of m_i^A then it reaches the node in distance 1 to the root$(h-1)\lceil c/n_{i+1} \rceil$ time slots later. Hence we can derive the inequality $h\lceil c/n_i \rceil < \delta_i + (h-1)\lceil c/n_{i+1} \rceil$ and the claim follows. □

Since a smaller message moves at at lower speed, a message m_{i+1} containing n_{i+1} packets can never catch up a message m_i with n_{i+1} or more packets, even if there is only one time slot between their departure and arrival.

We start analyzing input sequences where $\mathcal{AGG}$ does not merge any messages at inner nodes. We then show that merge operations cannot deteriorate the competitive ratio.

Lemma 5.10. *For sequences where $\mathcal{AGG}$ does not merge any messages, the competitive ratio of $\mathcal{AGG}$ is $O(\sqrt{h})$ on chain graphs.*

Proof. We pick a transmission of the optimal offline algorithm $\mathcal{OPT}$ and compare the cost accumulated for the packets within this message to the cost the online algorithm $\mathcal{AGG}$ incurs for the same packets. We show that the competitive ratio for each such transmission is in $O(\sqrt{h})$ by proving an upper bound on the cost for the online algorithm $\mathcal{AGG}$ followed by a lower bound on the cost the optimal algorithm accrues.

Let us consider the case where $\mathcal{OPT}$'s message m^O consists of packets distributed over z messages m_i^A of the online algorithm $\mathcal{AGG}$. In other words,

we assume that for a certain number of packets, the online algorithm sends messages m_i^A where $i \in \{1, \ldots, z\}$ and $\mathcal{OPT}$ sends one message m^O. If there are any packets in the first or the last message of $\mathcal{AGG}$ that the optimal algorithm $\mathcal{OPT}$ does not include in m^O, we ignore them when computing the delay and communication cost for $\mathcal{OPT}$. Let n_i denote the number of packets in message m_i^A. For $i = 1$ or $i = z$ we set n_i to the number of packets that are in m_i^A and in m^O .

When sending the z messages individually to the root, $\mathcal{AGG}$ incurs a transmission cost of $cc^{\mathcal{AGG}} = zhc$ and a delay cost of $dc^{\mathcal{AGG}} < 2zhc$.

After having bounded the cost for $\mathcal{AGG}$ we now turn to the minimum cost an offline algorithm faces. Clearly, the transmission of m^O entails a communication cost of $cc^{\mathcal{OPT}} = ch$ for $\mathcal{OPT}$. As a next step we want to determine the minimum delay cost accumulated by the z messages m_i^A, which are merged into message m^O by the optimal algorithm. Without loss of generality, we assume that for each message m_i^A, $1 \leq i \leq z$, all n_i packets arrive simultaneously, since $\mathcal{OPT}$ would clearly incur higher delay cost if packets arrived dispersed over time.

We consider two cases: a) the minimum delay cost if the majority of messages is large, i.e., contains more than $c/2$ packets, and b) the minimum delay cost if the majority of the packets is small, i.e., consists of at most $c/2$ packets. We show that in both cases the delay cost of the optimal algorithm is in $\Omega(cz^2)$.

a) If more than $z/2$ messages contain more than $c/2$ packets, we ignore all other messages and focus on the messages with $n_i > c/2$. Let δ_j denote the number of time slots between the arrival of message m_j^A and the arrival of m_{j+1}^A. Assume there are y messages with $n_i > c/2$. They are responsible for a total delay cost of at least

$$dc^{\mathcal{OPT}} > \sum_{i=1}^{y} n_i \sum_{j=i}^{y} \delta_j.$$

Even when $\delta_j = 1$ for all j, this sum amounts to

$$dc^{\mathcal{OPT}} > \sum_{i=1}^{y} n_i \sum_{j=i}^{y} \delta_j \geq \sum_{i=1}^{y} \frac{c}{2}(y - i) \in \Omega(y^2 c).$$

Thus this sum is in $\Omega(z^2 c)$, because $y > z/2$.

b) If the majority of messages m_i^A contain at most $c/2$ packets, we ignore the other messages. For simplicity's sake we let z denote the number of messages with $n_i \leq c/2$. Let λ_i denote the time period between the arrival of the n_i packets and the time slot when the corresponding message m_i^A departs from the leaf. Moreover, let δ_i be the time period after the message has departed from the leaf until the next set of n_{i+1} packets arrives. See Figure 5.5 for an example.

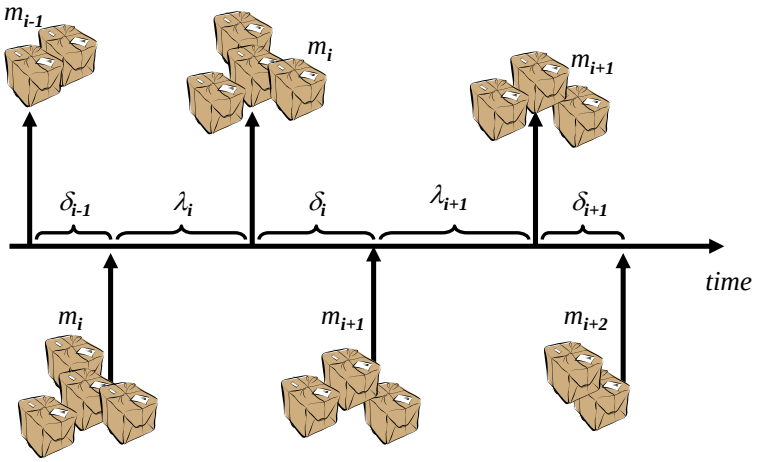


Figure 5.4: A sequence of message arrivals and departures for $\mathcal{AGG}$ at a leaf. The message m_i^A is forwarded after δ_i time slots at the leaf. δ_i time slots later the next message arrives.

The total sum of the delay cost the $\mathcal{OPT}$ accumulates at the leaf for all messages m_i^A is given by

$$dc^{\mathcal{OPT}} = \sum_{i=1}^{z-1} n_i \sum_{j=i}^{z-1} (\lambda_j + \delta_j) \,.$$

The following facts enable us to derive a bound for the delay cost the optimal algorithm incurs by waiting until the packets of all messages m_i^A have arrived at the leaf:
(i) If the number of packets between two messages differs by a factor of at most two, the time they spend at the leaf differs by the same factor.
(ii) If the difference between the size of two messages is larger than a factor of two, we can apply Lemma 5.9 to compute the minimum number of time slots between their arrival that ensures that a larger message cannot catch up with a smaller message on their path to the root. It holds that if $2n_i < n_j,\ j > i$

then the number of time slots m_j^A arrives later than m_i^A is at least

$$\begin{aligned} h\left\lceil \frac{c}{n_i} \right\rceil - (h-1)\left\lceil \frac{c}{n_j} \right\rceil &\geq h\frac{c}{n_i} - (h-1)\left(\frac{c}{n_j}+1\right) \\ &> \frac{2hc-(h-1)(c-2n_i)}{2n_i} \\ &> \frac{hc}{4n_i}, \end{aligned}$$

as $n_i < c/4$ due to our assumption that the largest message contains at most $c/2$ packets and no messages are merged after they leave the leaf node. More formally, the following holds.

Fact 5.11. *(i) If* $2n_i > n_j,\ j > i$ *then* $\lambda_j = \lceil c/n_j \rceil > c/2n_i \geq \lambda_i/2$. *(ii) If* $2n_i < n_j,\ j > i$ *then* $\sum_{k=i}^{j-1} \lambda_k + \delta_k > \frac{hc}{4ni}$.

We construct a binary vector a where $a_i = 1$ if there exists a message m_j^A with $j > i$ and $2n_i < n_j$. Using this vector we can rewrite the delay cost $dc^{\mathcal{OPT}}$ the optimal algorithm incurs at the leaf.

$$\begin{aligned} dc^{\mathcal{OPT}} &= \sum_{i=1}^{z-1} n_i \sum_{j=i}^{z-1} (\lambda_j + \delta_j) \\ &> \sum_{i=1}^{z-1} n_i \left(a_i \frac{hc}{4n_i} + (1-a_i) \sum_{j=i}^{z-1} \frac{c}{2n_i} \right) \qquad (5.1) \\ &> \sum_{i=1}^{z-1} \left(\frac{a_i hc}{4} + \frac{(1-a_i)(z-1-i)c}{2} \right) \\ &= \frac{c}{4} \sum_{i=1}^{z-1} (a_i(h-2z+2+2i) + 2z - 2 - 2i)\,. \qquad (5.2) \end{aligned}$$

In (5.1), we used the Fact 5.11 to bound $\sum_{j=i}^{z-1} \lambda_j + \delta_j$. If $a_i = 1$, we apply Fact 5.11(ii) and the sum is replaced by number of time slots until the large message responsible for $a_i = 1$ arrives, in the other case, we have a lower bound for λ_j due to Fact 5.11(i).

If $h > z - 2$ the sum in (5.2) is minimized if $a_i = 0$ for all messages. In this case it holds that

$$\begin{aligned} dc^{\mathcal{OPT}} &> \frac{c}{4} \sum_{i=1}^{z-1} (2z - 2 - 2i) \\ &= \frac{c}{4}(z-1)(2z-2-z) \\ &> \frac{c}{4}(z-2)^2 \\ &\in \Omega(cz^2). \end{aligned}$$

If $h < z-2$, then $\sum_{i=1}^{z-1} a_i(h-2z+2+2i)$ can be negative. This sum is minimized if the messages with $a_i = 1$ occur first. Let the number of these messages be y. In this case, we have to consider $\sum_{i=1}^{y} h-2z+2+2i = y(h-2z+3+y)$, which reaches its minimum of $-(1/4)(2z-h-3)^2$ for $y = z-h/2-3/2$. Consequently,

$$\begin{aligned} dc^{\mathcal{OPT}} &> \frac{c}{4}\sum_{i=1}^{z-1}(a_i(h-2z+2+2i)+2z-2-2i) \\ &> -\frac{c}{16}(2z-h-3)^2+\frac{c}{4}(z-2)^2 \\ &\in \Omega(cz^2). \end{aligned}$$

Hence $dc^{\mathcal{OPT}}$ is in $\Omega(cz^2)$ in both cases a) and b). We can conclude that combining the delay and communication cost for $\mathcal{OPT}$ yields the desired upper bound of the competitve ratio.

$$\rho = \frac{cost^{\mathcal{AGG}}}{cost^{\mathcal{OPT}}} \in O\left(\frac{zhc}{cz^2+ch}\right) = O\left(\min\left(\frac{h}{z}, z\right)\right) = O\left(\sqrt{h}\right).$$

□

We now investigate what happens if there are fewer time slots between packet arrivals, i.e., situations where merge operations can occur. To this end, we consider a sequence of packet arrivals without merge operations and compare it to the same sequence where the number of time slots between consecutive arrivals is reduced. More precisely, let $S = \{(n_1,t_1),(n_2,t_2),\ldots\}$ denote a sequence of packet arrivals, where (n_i,t_i) describes the arrival of n_i packets in time slot t_i. We compute the difference between the cost accumulated for sequences $S = \{(n_1,t_1),(n_2,t_2),\ldots\}$ without merged messages and $S' = \{(n'_1,t'_1),(n'_2,t'_2),\ldots\}$, where for all $i > 0$ it holds that $n_i = n'_i$ and $t_{i+1} - t_i \geq t'_{i+1} - t'_i$.

Lemma 5.12. *Consider a transmission of $\mathcal{OPT}$ in the sequence S' and assume that $\mathcal{AGG}$ merges μ_i messages into one message at hop distance i from the leaf. Compared to a sequence S where no messages are merged, $\mathcal{AGG}$ can reduce its cost by at least*

$$\Omega(\mu_i \cdot c \cdot (h-i)).$$

Proof. If the μ_i messages are sent to the root individually, the communication cost of the online algorithm is $\mu_i(h-i)c$ for the transmissions from the node at distance i to the root node. Merging theses messages reduces the cost for the transmission to $(h-i)c$. Consequently the difference is in $\Omega(\mu_i \cdot c \cdot (h-i))$. □

Lemma 5.13. *Consider a transmission of $\mathcal{OPT}$ in the sequence S' and assume that $\mathcal{AGG}$ merges μ_i messages into one message at hop distance i from the leaf. Compared to a sequence S where no messages are merged, $\mathcal{OPT}$ can reduce its cost by at most*

$$O(\mu_i \cdot c \cdot (h-i)).$$

Proof. Consider the μ_i messages that $\mathcal{AGG}$ merges at node i. We denote the size of the messages under scrutiny by $n_1, \ldots, n_{\mu_i}$. Note that a messages of size n_j cannot catch up with a message of greater size $n_j > n_i$ unless it merges with another message first. This implies, that in order to have μ_i messages merging at distance i from the leaf, the message size of $\mathcal{AGG}$'s messages has to be strictly monotonically increasing.

In the following, let the superscript s identify variables considered in the scenario where $\mathcal{AGG}$ does not merge any messages, and the superscript m identify variables in the other scenario. Let $\lambda_l + \delta_l$ denote the time period between two consecutive arrival time slots of the set of n_l and the set of n_{l+1} packets, where $\lambda_l = \lceil c/n_l \rceil$ is the number of time slots the l^{th} message spends at the leaf δ_l is the number of time slots between m_l^A's departure and m_{l+1}^A's arrival that ensure that m_{l+1}^A does not catch up (too early) with m_l^A. For sequences where $\mathcal{AGG}$ does not merge any packets, $\mathcal{OPT}$'s delay cost is given by

$$dc^s = \sum_{l=1}^{\mu_i - 1} \sum_{j=l}^{\mu_i - 1} n_l(\delta_j^s + \lambda_j).$$

In order to guarantee that consecutive messages are not merged by $\mathcal{AGG}$ we can apply Lemma 5.9 to determine $\delta_l^s = \max(\kappa_l^s, 0)$, where $\kappa_l^s = h\lceil c/n_l \rceil - (h-1)\lceil c/n_{l+1} \rceil - \lceil c/n_l \rceil$. Since $n_l < n_{l+1}$ we know that $\kappa_l^s < h(\lceil c/n_l \rceil - \lceil c/n_{l+1} \rceil)$ which is never negative. Note that this entails that

$$\sum_{j=l}^{\mu_i - 1} \delta_j^s \leq \lceil hc/n_l \rceil - \lceil hc/n_{\mu_i} \rceil.$$

If we assume the merge operation of μ_i messages to happen at depth $h-i$ we can compute a lower bound for the shortened time period between two messages. Observe that in order to ensure that messages do not merge too early, it must hold that $\delta_l^m > \max(\kappa_l^m, 0)$, where $\kappa_l^m = i\lceil c/n_l \rceil - (i-1)\lceil c/n_{l+1} \rceil - \lceil c/n_l \rceil$. Due to $n_l < n_{l+1}$, it holds that $\kappa_l^m \geq (i-1)(\lceil c/n_l \rceil - \lceil c/n_{l+1} \rceil)$ and thus

$$\sum_{j=l}^{\mu_i - 1} -\delta_j^m \geq \lceil (i-1)c/n_{\mu_i} - \lceil ic/n_l \rceil \rceil.$$

Combining the above implies that $\mathcal{OPT}$ can reduce its delay cost by at most

$$\begin{aligned}
\Delta dc^{\mathcal{OPT}} &= dc^s - dc^m \\
&\leq \sum_{l=1}^{\mu_i-1} n_l \sum_{j=l}^{\mu_i-1} \delta_j^s - \sum_{l=1}^{\mu_i-1} n_l \sum_{j=l}^{\mu_i-1} \delta_j^m \\
&= \sum_{l=1}^{\mu_i-1} n_l h \left(\left\lceil \frac{c}{n_l} \right\rceil - \left\lceil \frac{c}{n_{\mu_i}} \right\rceil \right) - (i-1) \left(\left\lceil \frac{c}{n_l} \right\rceil - \left\lceil \frac{c}{n_{\mu_i}} \right\rceil \right) \\
&\leq \sum_{l=1}^{\mu_i-1} n_l \left(h-i+1\right)) \left(\frac{c}{n_l} - \frac{c}{n_{\mu_i}} + 1 \right) \\
&= (\mu_i - 1)(h-i+1)(c+1).
\end{aligned}$$

□

Lemma 5.14. *Consider a sequence of packet arrivals on a chain graph satisfying the following conditions: $\mathcal{AGG}$ sends some messages individually to the root and performs one merge operation on the other messages. If $\mathcal{OPT}$ merges all these messages into one at the leaf then the competitive ratio is in $O(\min(\sqrt{h}, c))$.*

Proof. Let the number of messages $\mathcal{AGG}$ sends individually to the root be denoted by ν and the number of messages $\mathcal{AGG}$ merges at distance i from the leaf be denoted by μ. If the packets that form the μ messages $\mathcal{AGG}$ merges arrive before the ν messages sent to the root separately the claim follows directly from Lemmas 5.12 and 5.13. Otherwise we have to take the delay cost the ν messages can save since the to be merged messages can arrive closer to each other into account. Applying the Lemmas 5.12 and 5.13 leads to a total cost for $\mathcal{AGG}$ of less than $3(\nu hc + hc + \mu ic)$ and the total cost for $\mathcal{OPT}$ amounts to at least $\Omega\left(hc + \min(\nu^2 c, \nu hc) + \mu ic\right)$. As in the proofs above we can assume without loss of generality that $(\nu + \mu) \in \omega(\sqrt{h})$. If $\nu h < \nu^2$ the competitive ratio is in $O(1)$, otherwise the ratio is at most $O(\frac{\nu h + \mu}{h + \nu^2 + \mu})$. Assume ν to be lager than μ. This implies that $\nu > \sqrt{h}$ and hence the ratio is $O(\nu h/\nu^2) = O(\sqrt{h})$. If μ is larger than ν, we have a ratio of $O(hn\mu/(h + n\mu^2))$, which is $O(\sqrt{h})$. □

Hence, we have shown that the competitive ratio does not deteriorate for one occurrence of a merge operation. We can apply the above lemma and induction on the number of merge operations one message takes part in: Treat the node where a merge occurs as a new leaf node and consider the merge operations on the remaining chain until the root independently. For this network, the same arguments apply, and we can conclude that the cost the online and the offline algorithm save by merging does not inflict a change

in the asymptotic worst case competitive ratio. Hence it follows together with Theorem 5.3 that the competitive ratio is at most $O(\min(\sqrt{h}, c))$ on chain graphs. □

Lower Bound

We now show that $\mathcal{AGG}$ is asymptotically optimal for all oblivious online algorithms, i.e., we derive a lower bound for chain networks of $\Omega(\min(\sqrt{h}, c))$.

Theorem 5.15. *For chain networks the competitive ratio of any oblivious algorithm ALG is at least*

$$\rho = \frac{cost^{ALG}}{cost^{\mathcal{OPT}}} \in \Omega\left(\min(\sqrt{h}, c)\right).$$

Proof. Let *ALG* denote any oblivious online algorithm, and assume that packets arrive one-by-one: at time 0, a packet p arrives at the leaf node l. The next packet arrives at the leaf exactly when *ALG* sends the packet at l. Let w denote the time a packet waits at l, and observe that the same waiting time holds for all nodes on the way from the leaf to the root due to the oblivious nature of *ALG*. Thus, the total waiting time per packet is hw, and the communication cost is hc: $cost^{ALG} = hw + hc$. We now derive an upper bound on the optimal algorithm's cost for this sequence. We partition the packets into blocks of size $\sqrt{h}$, i.e., one message contains $\sqrt{h}$ packets. Thus, the communication cost per packet of this algorithm is $hc/\sqrt{h} = \sqrt{h}c$. The delay cost per message at the leaf is $\sum_{i=1}^{\sqrt{h}-1} iw \in \Theta(hw)$. In addition, each packet experiences one unit of delay per hop on the way up to the root. Thus, the optimal cost per packet is $cost^{\mathcal{OPT}} \leq \Theta(\sqrt{h}c + w\sqrt{h} + h)$. Therefore, it asymptotically holds for this sequence that

$$\rho \geq \frac{hc + hw}{\sqrt{h}c + w\sqrt{h} + h} = \frac{h(c + w)}{\sqrt{h}(c + w) + h}.$$

The lower bound follows from distinguishing three cases. If h and $c + w$ are asymptotically equivalent, this yields $\Omega(\sqrt{h})$. If h is asymptotically larger than $c + w$, the best oblivious algorithm choosing w as small as possible gives a lower bound of $\Omega(c)$. Finally, if $h < c + w$, the lower bound is in $\Omega(\sqrt{h})$, completing the proof.

□

Discussion

Our findings can be compared to the analysis presented in [63]. Their $\Omega(\sqrt{h})$ lower bound holds for arbitrary oblivious algorithms on *trees*. We have shown that this upper bound is too pessimistic, as general trees are inherently more difficult than chain topologies, and that the lower bound can be increased

to $\Omega(\min(h, c))$. For chain networks, we have generalized their result to arbitrary edge cost, yielding a lower bound of $\Omega(\min(\sqrt{h}, c))$, which is proved tight by $\mathcal{AGG}$.

5.6 Value-Sensitive Aggregation

There are many scenarios where, in contrast to the aggregation model considered above, the information to be delivered is not binary (e.g., event messages) but where arbitrary *value* aggregations need to be performed at the root. E.g., consider a set of sensor nodes measuring the temperature or oxygen levels at certain outdoor locations, and the root is interested to have up-to-date information on these measurements. In this case, larger value changes are more important and should be propagated faster to the root, e.g., such that an alarm can be raised soon in case of drastic environmental changes.

In the following, we consider a most simple topology: a network consisting of a leaf and a sink. Let the value measured by the leaf l at time t be l_t. We assume that the leaf node can only send the value it currently measures. The root node's latest value of node l at time t is denoted by r_t. We seek to minimize the following optimization function: $cost = M \cdot c + \sum_t |l_t - r_t|$ where M is the total number of message transmissions and c the cost per transmission, i.e., $M \cdot c$ is the total communication cost.

Typically, the values measured by a sensor node do not change arbitrarily, but there is a bound on the maximal change per time unit. In the following, we assume that the value measured by a node changes by at most Δ per time slot. Moreover, we assume that the sensor nodes can only measure discrete quantities, and that the difference between two measured values is at least ϵ.

Optimal Offline Algorithm for Link

There exists a simple optimal (offline) algorithm $\mathcal{OPT}$ which applies dynamic programming. $\mathcal{OPT}$ exploits the following optimal substructure: If we know the best sending times for all slots $t' < t$ given that a message is sent at time t', we can compute the optimal sending times assuming that $\mathcal{OPT}$ sends at time t.

Let the number of time slots under scrutiny be T. Note that, we only have to consider the time slots with value changes, because an optimal algorithm either sends a value immediately after it has changed or not until the value has changed again. Let time slot t_i denote the time slot when the i^{th} value change occurred and we set $t_0 = 0$. Determining all time slots with value changes requires iterating over all T time slots. To compute the minimum cost accumulated until time slot t_i, we consider each possible last transmission $j < i$ and add the inaccuracy cost which accrued at the root node between the two transmissions j and i.

Due to the arguments above, we can construct an array $OPT[\cdot]$ of size $\lambda+1$, where λ is the total number of value changes at the leaf node. We set $OPT[0] = 0$, as we assume that initially, the root stores the correct value. The remaining matrix entries are then computed as follows:

$$OPT[i] = \min_{j<i} \left(OPT[j] + c + |l_{t_i} - l_{t_j}| + \sum_{k=j+1}^{i-1} |l_{t_k} - l_{t_j}|(t_{k+1} - t_k) \right).$$

In time $O(\lambda^2)$ we construct a matrix $A[\cdot][\cdot]$ of size $(\lambda+1) \times (\lambda+1)$, where for all $1 \leq j, i \leq \lambda$ the entry $A_{i,j} = |l_{t_i} - l_{t_j}|(t_{i+1} - t_i)$. This allows us to compute any sum $\sum_{k=j+1}^{i-1} |l_{t_k} - l_{t_j}|(t_{k+1} - t_k)$ for $1 \leq i \leq j \leq \lambda$ in constant time. For a given i, $1 \leq i \leq \lambda$, computing $OPT[i]$ takes time $O(\lambda)$ using these precomputed values. Thus, we have the following theorem.

Theorem 5.16. *In a link network, the optimal aggregation strategy for T time slots can be computed in time $O(T+\lambda^2)$, where λ is the number of value changes at the leaf.*

If the input for the offline algorithm consists of a sequence of value changes and the time slots when they happened, the time complexity is in $O(\lambda^2)$.

Online Algorithm for Link

We propose the following online algorithm $\mathcal{AGG}$, which can be seen as a generalization of the algorithm presented in the previous section: The leaf l sends the value it currently measures if and only if $\sum_{t=\tau}^{T} |l_t - l_\tau| \geq c$, where τ is the last time l has transmitted its value and T is the current time.

For the analysis of $\mathcal{AGG}$, we consider the time periods between two transmissions of $\mathcal{AGG}$. For each such period, we can bound the competitive ratio yielding an overall competitive ratio. We first need the following helper lemma.

Lemma 5.17. *Let ρ be the competitive ratio of $\mathcal{AGG}$ when $\mathcal{AGG}$'s delay cost is c in each time period I, then $3\rho/2$ is an upper bound on the total competitive ratio.*

Proof. First observe that $\mathcal{AGG}$ can have a larger delay cost than c in a period, e.g., if in a time slot where the accumulated delay cost is $c-\varepsilon$ for an arbitrarily small $\varepsilon > 0$ there is a large value change of size Δ at the leaf. Hence, the online algorithm's delay in any period is at most $2(c-\epsilon)+\Delta$. Consider a period I where the online algorithm's delay cost is $2(c-\epsilon)+k$ for some $k \leq \Delta$. Compared to the case studied so far, $\mathcal{AGG}$'s delay cost will increase by at most $k+c-2\epsilon$. However, due to the large value change, we know that

the optimal algorithm's delay cost must increase by at least k as well. The new competitive ratio ρ' is hence

$$\begin{aligned}
\rho' &= \frac{CC'_{\mathcal{AGG}} + DC'_{\mathcal{AGG}}}{CC'_{\mathcal{OPT}} + DC'_{\mathcal{OPT}}} \\
&\leq \frac{cc^{\mathcal{AGG}} + dc^{\mathcal{AGG}} + k + c - 2\epsilon}{cc^{\mathcal{OPT}} + dc^{\mathcal{OPT}} + k} \\
&= \frac{cc^{\mathcal{AGG}} + dc^{\mathcal{AGG}}}{cc^{\mathcal{OPT}} + dc^{\mathcal{OPT}} + k} + \frac{k + c - 2\epsilon}{cc^{\mathcal{OPT}} + dc^{\mathcal{OPT}} + k} \\
&< \rho + \frac{c - 2\epsilon}{cc^{\mathcal{OPT}} + dc^{\mathcal{OPT}}} \\
&< \rho + \frac{c - 2\epsilon}{(cc^{\mathcal{AGG}} + dc^{\mathcal{AGG}})/\rho} \\
&< \rho + \frac{c - 2\epsilon}{2c/\rho} \\
&< \frac{3\rho}{2}.
\end{aligned}$$

□

Theorem 5.18. *The competitive ratio of $\mathcal{AGG}$ is*

$$\rho = \frac{cost^{\mathcal{AGG}}}{cost^{\mathcal{OPT}}} \in \Theta(c/\epsilon),$$

where c is the link cost and ϵ is the minimum difference between two values.

Proof. We first prove that $\rho \in O(c/\epsilon)$, and subsequently show that $\rho \in \Omega(c/\epsilon)$.

Proof for $\rho \in O(c/\epsilon)$: First, the ratio is computed under the assumption that the delay cost of $\mathcal{AGG}$ is exactly c; we then apply Lemma 5.17. We classify the possible types of periods I between two sending events of the online algorithm and consider them separately. Observe that for any period where the optimal algorithm $\mathcal{OPT}$ transmits, the competitive ratio is at most 2, as $\mathcal{OPT}$ has cost at least c and the online algorithm $\mathcal{AGG}$ has communication cost c and delay cost c. It remains to examine the situations where $\mathcal{OPT}$ does not send.

Assume that at the beginning of this period, $\mathcal{AGG}$ sends the value A_0, and at the end, it sends the value A_1. Furthermore we denote the optimal algorithm's value at the root at the beginning and at the end of this period by O_0 and O_1, respectively. We define $\delta_0 := |A_0 - O_0|$ and $\delta_1 := |A_1 - O_1|$ and examine all possible cases under the assumption that the delay cost of $\mathcal{AGG}$ is c for every period.

Case $\delta_0 = \delta_1 = 0$: If $\mathcal{OPT}$ has no transmission, it must have the same delay cost in this period as $\mathcal{AGG}$, because the initial and final values are the same, and hence $\rho_I \leq 2$.

Case $\delta_0 = 0, \delta_1 \neq 0$*:* If $\mathcal{OPT}$ has no transmission, it must have at least the same delay cost as $\mathcal{AGG}$ in I , thus $\rho_I \leq 2$.

Case $\delta_0 \neq 0, \delta_1 = 0$*:* If $\mathcal{OPT}$ has no transmission, it incurs at least delay cost δ_0 in the first time slot, as $\mathcal{AGG}$ sends the correct value at the beginning of the period. Hence, $\rho_I \leq 2c/\delta_0$.

Case $\delta_0 \neq 0, \delta_1 \neq 0$*:* In this case, $\mathcal{OPT}$ has delay cost δ_0 as well yielding $\rho_I \leq 2c/\delta_0$.

Thus, for each of these periods, $\rho_I \leq 2c/\delta_0$. It must hold that $\delta_0 > \epsilon$, and the claim follows by applying Lemma 5.17.

Lower bound: Consider the following sequence of values at the leaf node:

$$0, \quad \epsilon^{c/\epsilon}, \quad 0^{c/\epsilon-1}, \quad -\epsilon, \quad 0^{c/\epsilon-1}, \quad \epsilon, \quad 0^{c/\epsilon-1}, \quad -\epsilon, \quad \ldots$$

where α^β denotes that the value α remains for β time slots. Observe that for each subsequence $(0^{c/\epsilon-1}, -\epsilon, 0^{c/\epsilon-1}, \epsilon)$, 2ϵ is an upper bound on $\mathcal{OPT}$'s delay cost: It is the total delay cost if there are no transmissions at all in the entire sequence. In contrast, $\mathcal{AGG}$ has $2c$ delay cost plus two transmissions. Thus, neglecting the cost of the first time slot, we have $\rho \geq 4c/2\epsilon = 2c/\epsilon$. □

5.7 Concluding Remarks

In this chapter we have studied an online aggregation problem motivated by the increasing popularity of wireless sensor networks. When tackling this problem we face a trade-off between event notification time and transmission cost. We have studied a simple distributed algorithm which achieves a competitive ratio of $\Theta(\min(h, c))$ in case of general trees and $\Theta(\min(\sqrt{h}, c))$ in case of chains. In addition to binary event aggregation, we analyse a new model where the root is interested in the nodes' *values* (e.g., the measured temperature or humidity). In both settings there are many intriguing open questions for future research. E.g., the exploration of asynchronous models for the value-sensitive model, the study of non-oblivious algorithms, or algorithms which have a limited amount of information about the states of their neighbors, can yield deeper insights into the event aggregation problem and may also be useful in other applications. Moreover, it would be interesting to apply the competitive analysis framework [9] accounting for system inflicted nondeterminism in distributed settings to this problem.

The trade-off between delay and communication cost appears in various technical contexts, and it plays a role in the design of algorithms for wireless sensor networks and for Internet transfer protocols. Additionally, results in this area can carry over to surprisingly different problems. E.g., Papadimitriou et al. [91, 92] investigate the following optimization problem in organization theory: An organization is modeled as a tree where employees (leaves) receive messages to be sent to the boss (the root). The authors

observe that before it is possible to accept some communication, humans typically must do a "context switch" in order to process the message properly, and that the cost of these context switches becomes large if communications are not scheduled carefully. Concretely, the cost function consists of two components, one capturing the total number of messages sent along the tree, and the other one capturing the total delay incurred by the messages before they reach the root. This is very similar to the model we explored in this chapter. Papadimitriou et al. assume a Poisson process queuing model for the formal analysis of this dilemma of interruptions, but of course the bounds from this chapter hold in the organizational setting as well.

6

Conclusions and Outlook

In this part of the thesis we have investigated two areas that are critical to the successful operation of wireless networking: first, inadvertent and adversarial interference and second, the limited availability of energy. We discussed models and proposed algorithms for device discovery under Byzantine disruptions, power control, scheduling and an energy-latency trade-off.

The initial state in an ad hoc network is a collection of devices that are unaware of each other's presence. The very first step in building a network is thus, to find other devices. This device discovery procedure is a key step in configuring and optimizing the topology of the network. The shared nature of the communication medium in wireless networks induces their vulnerability to jamming attacks of various strengths. Of course, such a disruption of communication delays the discovery process. In Chapter 2, we presented algorithms that allow for fast device discovery degrading gracefully with the strength of the attacker. Our algorithms all have in common that they are simple and fully distributed, hence they are ideal candidates for energy and memory constrained sensor nodes.

Not only does interference constitute a challenge when establishing a network, it remains cumbersome in every phase of operation. Signals transmitted concurrently disturb each other, even if their respective senders are far away from each other. In order to still use the available bandwidth efficiently, we examined the scheduling problem and suggest algorithms with provable performance guarantees. In Chapter 4, we presented a power control and scheduling algorithm that remedies some of the drawbacks of previous approaches. In its current state, it is centralized and hence suitable to be employed in static networks with known traffic patterns only. Whether a distributed algorithm working in a manner similar to this algorithm exists, is an intriguing open question. Ideally, such a distributed worst-case efficient scheduling algorithm could lead to improved MAC-layer solutions, as combined power control and scheduling are crucial to a theoretical under-

standing of media access control problems. In Chapter 3, we considered networks where every node emits signals of the same power level. The main contribution of this chapter is a method of reducing a problem known to be NP-complete by constructing a geometric instance of the scheduling problem. The method consists in disposing nodes in the plane in a way that restricts the number of possible solutions and enforces the constraints of the NP-complete problem. We believe that this method of reduction can be adapted to prove other problems to be hard in the physical model. E.g., an exciting research direction is to analyze the complexity of the joint problem of power control and scheduling.

In Chapter 5 we have studied an online aggregation problem which can be regarded as a generalization of the classic *ski-rental problem* to trees. This generalization captures the trade-off between speed and energy prevalent in wireless sensor networks. We have analysed a simple algorithm that attempts to minimizes the delay until messages reach the root and features an economical use of energy by aggregating messages. Despite its straightforward approach, it achieves the best possible asymptotic competitive ratio in the class of oblivious, deterministic and fully distributed algorithms. Moreover, the algorithm fulfills its task without knowledge of the presence and the state of devices in its vicinity, and it does not base its decisions on previous events. Thus, apart from the efficiency criterion, this algorithm is attractive for practical applications as it poses minimal hardware requirements on sensor nodes (low memory and computational requirements). Obviously, a real network may impose several additional constraints that an useful algorithm has to take into account. This holds for all our results, e.g., even though our interference model is more realistic than many others, it still contains several simplifying assumptions and all our models entirely ignore the effect of mobility which is characteristic for the use of many wireless devices. Nevertheless, we believe that a solid theoretical underpinning is necessary for the design of any efficient system. We hope that, although some of our results are merely a small step towards a better understanding of the problems under scrutiny, our results will prove beneficial for future ad hoc and sensor networks.

Part II

Mechanism Design by Creditability

7

Manipulation in Games

7.1 Introduction

The quest for a deeper understanding of our world and its highly interconnected systems and processes often requires a huge amount of computational resources which can only be obtained by connecting thousands of computers. Similarly to agents in socio- economic systems, the computers in such networks often operate on a decentralized control regime, and represent various stake-holders with diverse objectives. Therefore, in addition to mere technical challenges, a system designer often has to take into account sociological and economic aspects as well when reasoning about protocols for maximizing system performance.

Game theory is a powerful tool for analyzing decision making in systems with autonomous and rational (or selfish) participants. It is used in a wide variety of fields such as biology, economics, politics, or computer science. A major achievement of game theory is the insight that networks of self-interested agents often suffer from inefficiency due to effects of selfishness. The concept of the price of anarchy allows to quantify these effects: The price of anarchy compares the performance of a distributed system consisting of selfish participants to the performance of an optimal reference system where all participants collaborate perfectly. If a game theoretic analysis of a distributed computing system reveals that the system has a large price of anarchy, this indicates that the protocol should be extended by a mechanism encouraging cooperation.

In many distributed systems, a mechanism designer cannot change the rules of interactions. However, she may be able to influence the agents' behavior by offering payments for certain outcomes. On this account, we consider a mechanism designer whose power is to some extent based on her monetary assets, primarily, though, on her creditability. That is, the players trust her to pay the promised payments. Thus, a certain subset of outcomes

is implemented in a given game if, by expecting additional non-negative payments, rational players will necessarily choose one of the desired outcomes. A designer faces the following optimization problem: How can the desired outcome be implemented at minimal cost? Surprisingly, it is sometimes possible to improve the performance of a given system merely by creditability, i.e., without any payments at all.

Chapter 8 presents several results for this problem. We propose algorithms for finding incentive compatible implementations of a desired set of outcomes, we show how a bankrupt mechanism designer can decide in polynomial time if a set of outcomes can be implemented at no cost at all, and an interesting connection to best response graphs is established. We propose and analyze efficient heuristic algorithms and demonstrate their performance. Furthermore, we extend our analysis for risk-averse behavior and study dynamic games where the mechanism designer offers payments in each round.

Whether a mechanism designer is prepared to invest the cost of implementing a desired outcome often depends on how much better than the original outcome the implemented outcome is. If the social welfare gain does not exceed the implementation cost, the mechanism designer might decide not to influence the game at all. In many games, however, manipulating the players' utility is profitable.

The following extension of the well-known prisoners' dilemma illustrates this phenomenon. Two bank robbers, both members of the *Al Capone clan*, are arrested by the police. The policemen have insufficient evidence for convicting them of robbing a bank, but they could charge them with a minor crime. Cleverly, the policemen interrogate each suspect separately and offer both of them the same deal. If one testifies to the fact that his accomplice has participated in the bank robbery, they do not charge him for the minor crime. If one robber testifies and the other remains silent, the former goes free and the latter receives a three-year sentence for robbing the bank and a one-year sentence for committing the minor crime. If both betray the other, each of them will get three years for the bank robbery. If both remain silent, the police can convict them for the minor crime only and they get one year each. There is another option, of course, namely to confess to the bank robbery and thus supply the police with evidence to convict both criminals for a four-year sentence (cf. G in Figure 7.1). A short game-theoretic analysis shows that a player's best strategy is to testify. Thus, the prisoners will betray each other and both get charged a three-year sentence.

Now assume that Mr. Capone gets a chance to take influence on his employees' decisions. Before they take their decision, Mr. Capone calls each of them and promises that if they both remain silent, they will receive money compensating for one year in jail,[1] and furthermore, if one remains silent and the other betrays him, Mr. Capone will pay the former money worth two years

[1] For this scenario, we presume that time really is money!

in prison (cf. V in Figure 7.1). Thus, Mr. Capone creates a new situation for the two criminals where remaining silent is the most rational behavior. Mr. Capone has saved his clan two years in jail.

Let us consider a slightly different scenario. After the police officers have made their offer to the prisoners, their commander-in-chief devises an even more promising plan. He offers each criminal to drop two years of the four-year sentence in case he confesses the bank robbery and his accomplice be-

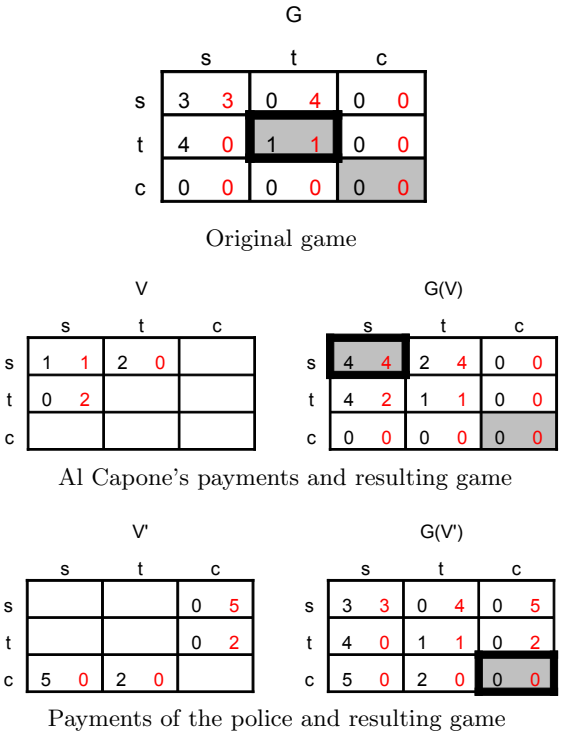


Figure 7.1: Extended prisoners' dilemma: G shows the prisoners' initial payoffs, where payoff values equal saved years. The first strategy is to remain silent (s), the second to testify (t) and the third to confess (c). Nash equilibria are colored gray, and non-dominated strategy profiles have a bold border. The left bimatrix V shows Mr. Capone's offered payments which modify G to the game $G(V)$. By offering payments V', the police implements the strategy profile (c, c). As $V_1(c, c) = V_2(c, c) = 0$, payments V' implement (c, c) for free.

trays him. Moreover, if he confesses and the accomplice remains silent they would let him go free and even reward his honesty with a share of the booty (worth going to prison for one year). However, if both suspects confess the robbery, they will spend four years in jail. In this new situation, it is most rational for a prisoner to confess. Consequently, the commander-in-chief implements the best outcome from his point of view without dropping any sentence and he increases the accumulated years in prison by two.

From Mr. Capone's point of view, implementing the outcome where both prisoners keep quiet results in four saved years for the robbers. By subtracting the implementation cost, the equivalent to two years in prison, from the saved years, we see that this implementation yields a benefit of two years for the Capone clan. We say that the *leverage* of the strategy profile where both prisoners play s is two. For the police, the leverage of the strategy profile where both prisoners play c is two, since the implementation costs nothing and increases the years in prison by two. Since implementing c reduces the players' gain, we say the strategy profile where both play c has a *malicious leverage* of two. In the described scenario, Mr. Capone and the commander-in-chief solve the optimization problem of finding the game's strategy profile(s) which bear the largest (malicious) leverage and therewith the problem of implementing the corresponding outcome at optimal cost.

Chapter 9 analyzes these problems' complexities and presents algorithms for finding the leverage of games for cautious and optimistic mechanism designers.

In the remainder of this chapter, we review related work and give an overview of our contributions, followed by an introduction of our model and some basic game theoretic definitions.

7.2 Related Work

The mathematical tools of game theory have become popular in computer science recently as they allow to gain deeper insights into the socio-economic complexity of today's distributed systems. Game theory combines algorithmic ideas with concepts and techniques from mathematical economics. Popular problems in computer science studied from a game theoretic point of view include *virus propagation* [10], *congestion* [27], *wireless spectrum auctions* [120], among many others.

The observation that systems often perform poorly in the presence of selfish players has sparked research for countermeasures [33, 80]. For example, Cole et al. [28, 29] have study how incentive mechanisms can influence selfish behavior in a routing system where the latency experienced by the network traffic on an edge of the network is a function of the edge congestion, and where the network users are assumed to selfishly route traffic on minimum-latency paths. They show that by pricing network edges the inefficiency

of selfish routing can always be eradicated, even for heterogeneous traffic in single-commodity networks. Wang et al. [111] present a unicast routing system for non-cooperative wireless networks which is socially efficient.

Typically, when a game-theoretic analysis reveals that a system may suffer from selfish behavior, appropriate countermeasures are taken in order to enforce a desired behavior (e.g.,[43]). As it is often infeasible for a mechanism designer to influence the rules according to which the players act in a distributed system, she has to resort to other measures. One way to manipulate the players' decisions is to offer them money for certain outcomes.

Monderer and Tennenholtz [81] show how creditability can be used to outwit selfish agents and influence their decisions. They consider a mechanism designer who cannot enforce behaviors and cannot change the system, and who attempts to lead agents to adopt desired behaviors in a given multi-agent setting. The only way the third party can influence the outcome of the game is by promising non-negative monetary transfers conditioned on the observed behavior of the agents. The authors demonstrate that the mechanism designer might be able to induce a desired outcome at very low cost. In particular, they prove that any pure Nash equilibrium of a game with complete information has a *zero-implementation*, i.e., it can be transformed into a dominant strategy profile at zero cost. Similar results hold for any given ex-post equilibrium of a frugal VCG mechanism. Moreover, the paper addresses the question of the hardness of computing the minimal implementation cost.

We extend [81] in various respects. We suggest several new algorithms, for instance a polynomial time algorithm for deciding whether a set of strategy profiles has a 0-implementation. We point out connections to graph-theoretic concepts and we generalize the theorem by Monderer and Tennenholtz on the cost of Nash equilibria. We correct their algorithm for computing an optimal exact implementation, and we provide evidence that their **NP**-hardness proof of deciding whether a k-implementation exists is wrong. [81] attends to pessimistic mechanism designers calculating with maximum possible payments for a desired outcome. In this thesis, we also consider less anxious mechanism designers taking the risk of high worst case cost if the expected cost is small. For the latter we prove that computing the optimal implementation cost is NP-hard in general. Regarding pessimistic mechanism designers we were less lucky, since Monderer and Tennenholtz' complexity results turned out to be wrong and we were not able to fix them. Therefore, we propose polynomial-time heuristic algorithms and evaluate their performance by simulations. Furthermore, we generalize the implementation concept to other game theoretic models. We examine players aiming at maximizing the average payoff and show how the mechanism designer can find such implementations. As another contribution, we consider the case of risk-averse players and the complexity of computing the optimal implementation, and we initiate the study of mechanism design by creditability in round based dynamic

games, where the mechanism can adapt the offered payments in every round. Furthermore, we introduce the concept of leverage, a measure for the change of behavior a mechanism design can inflict, taking into account the social gain and the implementation cost. Regarding the payments offered by the mechanism designer as some form of insurance, it seems natural that outcomes of a game can be improved at no cost. However, as a first contribution, we show that a malicious mechanism designer can in some cases even reduce the social welfare at no cost. Second, we present algorithms to compute both the beneficial as well as the malicious leverage, and provide evidence that several optimization problems related to the leverage are NP-hard.

To the best of our knowledge, this is the first work studying malicious mechanism designers which aim at influencing a game based primarily on their creditability. Other types of maliciousness have been studied before in various contexts, especially in cryptography, and it is beyond the scope of this thesis to provide a complete overview of this literature. Recently, the concept of BAR games [2] has been introduced which aims at understanding the impact of altruistic and malicious behavior in game theory. Moscibroda et al. [83] extend the virus inoculation game from [10] to comprise both selfish and malicious players. A similar model has recently been studied in the context of congestion games [13]. Our work is also related to *Stackelberg theory* [100] where a fraction of the entire population is orchestrated by a global leader. In contrast to our model, the leader is not bound to offer any incentives to follow her objectives. Finally, in the recent research thread of *combinatorial agencies* [12], a setting is studied where a mechanism designer seeks to influence the outcome of a game by contracting the players individually; however, as she is not able to observe the players' actions, the contracts can only depend on the overall outcome.

In recent years, many mechanism design results involving payments of money, stamps, points or similar objects of value have been proposed. In networking, the fundamental problem with these schemes is that relying on monetary transfers often imposes a high implementation barrier [55]. Therefore, researchers have started investigating mechanisms *without money*. Unfortunately, the fundamental *Arrow's Theorem* [8, 101] shows that the power of mechanisms without money is severely limited in general. However, the observation that computer systems typically have the ability to arbitrarily reduce service quality (e.g., by dropping messages or insert delays) has given raise to the study of how such "punishments" can increase the social gain beyond the loss inflicted [55]. In contrast to the non-negative payments examined in our work, they use negative incentives to steer the players to a desired behavior. Moscibroda and Schmid [82] analyze mechanisms without payments for a model similar to the one studied in this article.Their work can be regarded as an application of the theories devised in this article to the domain of throughput maximization.

7.3 Preliminaries and Model

Game Theory

A finite *strategic game* can be described by a tuple $G = (N, X, U)$, where $N = \{1, 2, \ldots, n\}$ is the set of *players* and each player $i \in N$ can choose a *strategy* (action) from the set X_i. The product of all the individual players' strategies is denoted by $X := X_1 \times X_2 \times \ldots \times X_n$. In the following, a particular outcome $x \in X$ is called *strategy profile* and we refer to the set of all other players' strategies of a given player i by $X_{-i} = X_1 \times \ldots \times X_{i-1} \times X_{i+1} \times \ldots \times X_n$. An element of X_i is denoted by x_i, and similarly, $x_{-i} \in X_{-i}$; hence x_{-i} is a vector consisting of the strategy profiles of x_i. Finally, $U = (U_1, U_2, \ldots, U_n)$ is an n-tuple of *payoff functions* (utilities), where $U_i : X \to \mathbb{R}$ determines player i's payoff arising from the game's outcome.

The *social gain* of a game's outcome is given by the sum of the individual players' payoffs at the corresponding strategy profile x, i.e. $gain(x) := \sum_{i=1}^{n} U_i(x)$. Let $x_i, x_i' \in X_i$ be two strategies available to Player i.

We say that x_i *dominates* x_i' iff $U_i(x_i, x_{-i}) \geq U_i(x_i', x_{-i})$ for every $x_{-i} \in X_{-i}$ and there exists at least one x_{-i} for which a strict inequality holds. x_i is the *dominant* strategy for player i if it dominates every other strategy $x_i' \in X_i \backslash \{x_i\}$. x_i is a *non-dominated* strategy if no other strategy dominates it. By $X^* = X_1^* \times \ldots \times X_n^*$ we will denote the set of non-dominated strategy profiles, where X_i^* is the set of non-dominated strategies available to the individual player i.

The set of *best responses* $B_i(x_{-i})$ for player i given the other players' actions is defined as $B_i(x_{-i}) := \{x_i | U_i(x_i, x_{-i}) = \max_{x_j \in X_i \backslash \{x_i\}} U_i(x_j, x_{-i})\}$. A *Nash equilibrium* is a strategy profile $x \in X$ such that for all $i \in N$, $x_i \in B_i(x_{-i})$, i.e., no player has an incentive to choose a different strategy unilaterally.

Mechanism Design by Creditability

Our model is based on the classic assumption that players are rational and always choose a non-dominated strategy. Additionally, we assume that players do not collude. We examine the impact of payments to players offered by a *mechanism designer* (an interested third party) who seeks to influence the outcome of a game. These payments are described by a tuple of non-negative payment functions $V = (V_1, V_2, \ldots, V_n)$, where $V_i : X \to \mathbb{R}^+$, i.e. the payments for player i depend on the strategy Player i selects as well as on the choices of all other players. Thereby, we assume that the players trust the mechanism designer to finally pay the promised amount of money, i.e., consider her trustworthy (*mechanism design by creditability*). The original game $G = (N, X, U)$ is modified to $G(V) := (N, X, [U + V])$ by these payments, where $[U + V]_i(x) = U_i(x) + V_i(x)$, that is, each player i obtains the pay-

ments of V_i in addition to the payoffs of U_i. The players' choice of strategies changes accordingly: Each player now selects a non-dominated strategy in $G(V)$. Henceforth, the set of non-dominated strategy profiles of $G(V)$ is denoted by $X^*(V)$. Observe that we have made two implicit assumptions: The mechanism designer can observe the actions chosen by the players and the players can determine the payoffs of all their strategies and compute the best strategy among them.

A *strategy profile set* – also called *strategy profile region* – $O \subseteq X$ of G is a subset of all strategy profiles X, i.e., a region in the payoff matrix consisting of one or multiple strategy profiles. Similarly to X_i and X_{-i}, we define $O_i := \{x_i | \exists x_{-i} \in X_{-i} \text{ s.t. } (x_i, x_{-i}) \in O\}$ and $O_{-i} := \{x_{-i} | \exists x_i \in X_i \text{ s.t. } (x_i, x_{-i}) \in O\}$.

The mechanism designer's main objective is to force the players to choose a certain strategy profile or a set of strategy profiles, without spending too much. We study two kinds of implementation costs: *worst-case implementation cost* and *uniform implementation cost*.

First, we will consider a pessimistic scenario where the mechanism designer calculates with the maximum possible payments for a desired outcome (*worst-case implementation cost*). For a desired strategy profile set O, we say that payments V *implement* O if $\emptyset \subset X^*(V) \subseteq O$. V is called (worst-case) *k-implementation* if, in addition $V(x) \leq k$, $\forall x \in X^*(V)$. That is, the players' non-dominated strategies are within the desired strategy profile, and the payments do not exceed k for any possible outcome. Moreover, V is an *exact k-implementation* of O if $X^*(V) = O$ and $V(x) \leq k \ \forall x \in X^*(V)$.

The *cost* $k(O)$ of implementing O is the lowest of all non-negative numbers q for which there exists a q-implementation. If an implementation meets this lower bound, it is optimal, i.e., V is an *optimal implementation* of O if V implements O and $\max_{x \in X^*(V)} V(x) = k(O)$. The cost $k^*(O)$ of implementing O exactly is the smallest non-negative number q for which there exists an exact q-implementation of O. V is an *optimal exact implementation* of O if it implements O exactly and requires cost $k^*(O)$. The set of all implementations of O will be denoted by $\mathcal{V}(O)$, and the set of all exact implementations of O by $\mathcal{V}^*(O)$.

A strategy profile set $O = \{z\}$ of cardinality one – consisting of only one strategy profile – is called a *singleton*. Clearly, for singletons it holds that non-exact and exact k-implementations are equivalent. For simplicity's sake we often write z instead of $\{z\}$. Observe that only subsets of X which are in $2^{X_1} \times 2^{X_2} \times \ldots \times 2^{X_n}$, i.e., the Cartesian product of subsets of the players' strategies, can be implemented exactly. We call such a subset of X a *convex strategy profile set*.[2] In conclusion, we have the following definitions for the worst-case implementation cost:

[2]These sets define a convex area in the n-dimensional hyper-cuboid, provided that the strategies are depicted such that all o_i are next to each other.

Definition 7.1 (Worst-Case Cost and Exact Worst-Case Cost)**.** *A strategy profile set O has* worst-case implementation cost $k(O) := \min_{V \in \mathcal{V}(O)} \{\max_{z \in X^*(V)} V(z)\}$. *A strategy profile set O has* exact worst-case implementation cost $k^*(O) := \min_{V \in \mathcal{V}^*(O)} \{\max_{z \in X^*(V)} V(z)\}$.

The assumption that the cost of an implementation V is equal to the cost of the strategy profile in $X^*(V)$ with the *highest* payments is pessimistic. We can also consider a less anxious mechanism designer who takes the risk of high worst case cost if the expected cost is small. If players only know their own utilities, assuming them to select one of their non-dominated strategies uniformly at random, is a first simple model an optimistic mechanism designer might apply. We define the uniform cost of an implementation V as the *average* of all strategy profiles' possible cost in $X^*(V)$. Thus we assume all non-dominated strategy profiles $x \in X^*(V)$ to have the same probability.

Definition 7.2 (Uniform Cost and Exact Uniform Cost)**.** *A strategy profile set O has* uniform implementation cost $k_{UNI}(O) := \min_{V \in \mathcal{V}(O)} \{\varnothing_{z \in X^*(V)} V(z)\}$ *where $\varnothing$ is defined as* $\varnothing_{x \in X} f(x) := 1/|X| \cdot \sum_{x \in X} f(x)$. *A strategy profile set O has* exact uniform implementation cost $k^*_{UNI}(O) := \min_{V \in \mathcal{V}^*(O)} \{\varnothing_{z \in X^*(V)} V(z)\}$.

(Malicious) Leverage

Mechanism designers can implement desired outcomes in games at certain cost. This raises the question for which games it makes sense to take influence at all. This part examines two diametrically opposed kinds of interested third parties, the first one being *benevolent* towards the participants of the game, and the other being *malicious*. While the former is interested in increasing a game's social gain, the latter seeks to minimize the players' welfare. We define a measure indicating whether the mechanism of implementation enables them to modify a game in a favorable way such that their gain exceeds the manipulation's cost. We call these measures the *leverage* and *malicious leverage*, respectively. Note that in the following, we will often write "(malicious) leverage" to describe both leverage and malicious leverage.

As the concept of leverage depends on the implementation cost, we examine the *worst-case* and the *uniform* leverage. The worst-case leverage is a lower bound on the mechanism designer's influence: We assume that without the additional payments, the players choose a strategy profile in the original game where the social gain is maximal, while in the modified game, they select a strategy profile among the newly non-dominated profiles where the difference between the social gain and the mechanism designer's cost is minimized. The value of the leverage is given by the net social gain achieved by this implementation minus the amount of money the mechanism designer had to spend. For malicious mechanism designers we have to invert signs and swap max and min. Moreover, the payments made by the mechanism

designer have to be subtracted twice, because a malicious mechanism designer considers money received by the players to be a loss. The leverage and malicious leverage of a strategy profile set O are defined as follows.

Definition 7.3 (Worst-Case (Malicious) Leverage)**.** *Let*

$$lev(O) := \max_{V \in \mathcal{V}(O)} \{ \min_{z \in X^*(V)} \{U(z) - V(z)\} \} - \max_{x^* \in X^*} U(x^*)$$

and

$$mlev(O) := \min_{x^* \in X^*} U(x^*) - \min_{V \in \mathcal{V}(O)} \{ \max_{z \in X^*(V)} \{U(z) + 2V(z)\} \}.$$

The leverage *and* malicious leverage *of a strategy profile set O are $LEV(O) := \max\{0, lev(O)\}$ and $MLEV(O) := \max\{0, mlev(O)\}$, respectively.*

Observe that according to our definitions, leverage values are always non-negative, as a mechanism designer has no incentive to manipulate a game if she will lose money. If the desired set consists only of one strategy profile z, i.e., $O = \{z\}$, we will speak of the *singleton* leverage. Similarly to the (worst-case) leverage, we can define the uniform leverage for less anxious mechanism designers.

Definition 7.4 (Uniform (Malicious) Leverage)**.** *Let*

$$lev_{UNI}(O) := \max_{V \in \mathcal{V}(O)} \{ \operatorname*{⌀}_{z \in X^*(V)} (U(z) - V(z)) \} - \operatorname*{⌀}_{x^* \in X^*} U(x^*)$$

and

$$mlev_{UNI}(O) := \operatorname*{⌀}_{x^* \in X^*} U(x^*) - \min_{V \in \mathcal{V}(O)} \{ \operatorname*{⌀}_{z \in X^*(V)} \{U(z) + 2V(z)\} \}.$$

The uniform leverage *and* malicious uniform leverage *of a strategy profile set O are $LEV_{UNI}(O) := \max\{0, lev_{UNI}(O)\}$ and $MLEV_{UNI}(O) := \max\{0, mlev_{UNI}(O)\}$, respectively.*

We define the *exact (uniform) leverage* $LEV^*(O)$ and the *exact (uniform) malicious leverage* $MLEV^*(O)$ by simply changing $\mathcal{V}(O)$ to $\mathcal{V}^*(O)$ in the definition of $LEV_{(UNI)}(O)$ and $MLEV_{(UNI)}(O)$. Thus, the exact (uniform) (malicious) leverage measures a set's leverage if the interested party may only promise payments which implement O exactly.

8

k-Implementations

This chapter attends to the problem of a mechanism designer seeking to influence the outcome of a strategic game based on her creditability. The mechanism designer offers additional payments to the players depending on their mutual choice of strategies in order to steer them to certain decisions. Of course, the mechanism designer aims at spending as little as possible and yet implementing her desired outcome. We present several algorithms and complexity results for this optimization problem both for singleton target strategy profiles and target strategy profile regions. After considering the implementation cost of singleton in the first section, we discuss how a greedy mechanism designer that does not want to spend any money can influence games. The third section examines a pessimistic scenario where the mechanism designer calculates with the maximum possible payments for a desired outcome (*worst-case implementation cost*) followed by the fourth section where we assume a more optimistic mechanism designer that bases her decision on the expected payments due if an outcome is chosen uniformly at random out of the implemented set of outcomes. We conclude our chapter on k-implementations by a brief excursion on other rationality models the players might adopt.

8.1 Singletons

In order to become familiar with the concepts and the notation of mechanism design by creditability, we begin by considering the smallest implementable unit of a game: a singleton. Note that for singletons the notions of worst case implementation cost and uniform implementation cost coincide. In [81], Monderer and Tennenholtz characterize the implementation cost of a singleton as follows.

Theorem 8.1 ([81]). *Let $G = (N, X, U)$ be a game with at least two strategies for every player. Every strategy profile z has an implementation V, and moreover its implementation cost amounts to*

$$k(z) = \sum_{i=1}^{n} \max_{x_i \in X_i} \left(U_i(x_i, z_{-i}) - U_i(z_i, z_{-i})\right).$$

Proof. Assume V implements z. In this case, the only non-dominated strategy profile in the modified game $G(V)$ is z, i.e., $X^*(V) = z$. Thus V has to ensure that for every $i \in N$ it holds that z_i is the only non-dominated strategy. Formally, the following condition has to be met:

$$V_i(z_i, x_{-i}) + U_i(z_i, x_{-i}) \geq U_i(x_i, x_{-i}) \text{ for every } x_i \in X_i \text{ and } x_{-i} \in X_{-i}.$$

That is

$$V_i(z_i, x_{-i}) \geq \max_{x_i \in X_i} \left(U_i(x_i, x_{-i}) - U_i(z_i, x_{-i})\right) \text{ for every } x_{-i} \in X_{-i}.$$

V satisfying this inequality does not guarantee that it is the only non-dominated strategy available to player i. We remedy this by increasing V's payments for every strategy profile where $x_{-i} \neq z_{-i}$ (here we use our assumption that every player has at least two strategies). Note that this does not change the cost of our implementation, as the mechanism designer is only charged for the payments in $X^*(V) = z$. The payments for strategy profiles where $x_i \neq z_i$ do not have to be modified to reach our goal. Hence the statement of the theorem follows by constructing V such that

$$V_i(x_i, x_{-i}) = \begin{cases} 0 & \text{if } x_i \neq z_i \\ \max\limits_{x_i \in X_i} \left(U_i(x_i, x_{-i}) - U_i(z_i, x_{-i})\right) & \text{if } x_i = z_i, x_{-i} = z_{-i} \\ \max\limits_{x_i \in X_i} \left(U_i(x_i, x_{-i}) - U_i(z_i, x_{-i})\right) + 1 & \text{if } x_i = z_i, x_{-i} \neq z_{-i} \end{cases}$$

and summing up the payments for all players. □

Observe that z constitutes a Nash equilibrium if and only if for every player $i \in N$, $\max_{x_i \in X_i}(U_i(x_i, z_{-i}) - U_i(z_i, z_{-i})) = 0$ for every $x_{-i} \in X_{-i}$. Hence the following characterization of Nash equilibria can be deduced from Theorem 8.1.

Corollary 8.2 ([81]). *Let $G = (N, X, U)$ be a game with at least two strategies for every player. A strategy profile z is a Nash equilibrium if and only if z has a 0-implementation.*

This corollary entails that some outcomes than be implemented without spending anything. In the next section, we demonstrate that not only singletons, but also strategy profile sets can feature 0-implementation. Moreover, strategy profile sets exhibit the nice property, that we can be determine efficiently if they can be implemented for free.

8.2 Bankrupt Mechanism Designers

Imagine a mechanism designer who is broke. At first sight, it seems that without any money, she will hardly be able to influence the outcome of a game. However, this intuition ignores the power of creditability: a game can have 0-implementable regions. As for singletons, the definitions of worst case and uniform implementation cost coincide for 0-implementations.

Let V be an exact implementation of O with exact cost $k^*(O)$. It holds that if $k^*(O) = 0$, V cannot contain any payments larger than 0 in O. Consequently, for a region O to be 0-implementable exactly, any strategy s outside O_i must be dominated within the range of O_{-i} by a o_i, or there must be one o_i for which no payoff $U_i(s, o_{-i})$ is larger than $U_i(o_i, o_{-i})$. In the latter case, the strategy o_i can still dominate s by using a payment $V(o_i, x_{-i})$ with $x_{-i} \in X_{-i} \backslash O_{-i}$ outside O. Note that this is only possible under the assumption that $O_{-i} \subset X_{-i}$ $\forall i \in N$.

$\mathcal{ALG}_{bankrupt}$ (cf. Algorithm 8.1) describes how a bankrupt designer can decide in polynomial time whether a certain region is 0-implementable. It proceeds by checking for each player i if the strategies in $X_i^* \backslash O_i$ are dominated or "almost" dominated within the range of O_{-i} by at least one strategy inside O_i. If there is one strategy without such a dominating strategy, O is not 0-implementable exactly. On the other hand, if for every strategy $s \in X_i^* \backslash O_i$ such a dominating strategy is found, O can be implemented exactly without expenses.

Algorithm 8.1 Exact 0-Implementation ($\mathcal{ALG}_{bankrupt}$)

Input: Game G, convex region O with $O_{-i} \subset X_{-i}$ $\forall i$
Output: $\top$ if $k^*(O) = 0$, $\bot$ otherwise

```
 1: compute X*;
 2: for all i ∈ N do
 3:    for all s ∈ X_i* \ O_i do
 4:       dZero := ⊥;
 5:       for all o_i ∈ O_i do
 6:          b := ⊤;
 7:          for all o_{-i} ∈ O_{-i} do
 8:             b := b ∧ (U_i(s, o_{-i}) ≤ U_i(o_i, o_{-i}));
 9:          od
10:          dZero := dZero ∨ b;
11:       od
12:       if ¬ dZero then
13:          return ⊥;
14:       fi
15:    od
16: od
17: return ⊤;
```

Theorem 8.3. *Given a convex strategy profile region O where $O_{-i} \subset X_{-i}\ \forall i$, Algorithm $\mathcal{ALG}_{bankrupt}$ decides whether O has an exact 0-implementation in time $\mathrm{O}\big(n\,|X|^2\big)$.*

Proof. $\mathcal{ALG}_{bankrupt}$ is correct because it checks for each yet to be dominated strategy $s \in X_i^* \backslash O_i$ whether it can be dominated by one $o_i \in O_i$ at zero cost. This is the property that makes O exactly 0-implementable. Computing X^* takes time $\mathrm{O}\big(n\,|X|^2\big)$. All other costs are asymptotically negligible. □

Best Response Graphs

Best response strategies maximize the payoff for a player given the other players' decisions. For now, let us restrict our analysis to games where the sets of best response strategies consist of only one strategy for each x_{-i} $\forall i \in N$. Given a game G, we construct a directed *best response graph* $\mathcal{G}_G$ with vertices v_x for strategy profiles $x \in X$ iff x is a best response for at least one player, i.e., if $\exists i \in N$ such that $x_i \in B_i(x_{-i})$. There is a directed edge $e = (v_x, v_y)$ iff $\exists i \in N$ such that $x_{-i} = y_{-i}$ and $\{y_i\} = B_i(y_{-i})$. In other words, an edge from v_x to v_y, indicates that it is better to play y_i instead of x_i for a player if for the other players' strategies $x_{-i} = y_{-i}$. A strategy profile region $O \subset X$ has a *corresponding subgraph* $\mathcal{G}_{G,O}$ containing the vertices $\{v_x | x \in O\}$ and the edges which both start and end in a vertex of the subgraph. We say $\mathcal{G}_{G,O}$ has an *outgoing edge* $e = (v_x, v_y)$ if $x \in O$ and $y \notin O$. Note that outgoing edges are not in the edge set of $\mathcal{G}_{G,O}$. Clearly, it holds that if a singleton x's corresponding subgraph $\mathcal{G}_{G,\{x\}}$ has no outgoing edges then x is a *Nash equilibrium*. More generally, we make the following observation.

Theorem 8.4. *Let G be a game and $|B_i(x_{-i})| = 1\ \forall i \in N, x_{-i} \in X_{-i}$. If a convex region O has an exact 0-implementation, then the corresponding subgraph $\mathcal{G}_{G,O}$ in the game's best response graph has no outgoing edges.*

Proof. Let V be an exact 0-implementation of O. Note that $V(o) = 0\ \forall o \in O$, otherwise the cost induced by V are larger than 0. Assume for the sake of contradiction that $\mathcal{G}_{G,O}$ has an outgoing edge. Let $x \in O$ be a strategy profile for which its corresponding vertex v_x has an outgoing edge e to $v_y, y \in X \backslash O$. Since $V(x)$ is 0, $\mathcal{G}_{G(V),O}$ still has the same outgoing edge e. This means that for one Player j it is better to play strategy y_j in $G(V)$ than to play x_j given that $x_{-j} = y_{-j}$. Since y_j is not dominated by any strategy in O_j, Player j will hence choose also strategies outside O_j and therefore V is not a correct implementation of O, thus contradicting our assumption. □

In order to extend best response graphs to games with multiple best responses, we modify the edge construction as follows: In the general best response graph $\mathcal{G}_G$ of a game G there is a directed edge $e = (v_x, v_y)$ iff $\exists i \in N$ s.t. $x_{-i} = y_{-i}$, $y_i \in B_i(y_{-i})$ and $|B_i(y_{-i})| = 1$.

Corollary 8.5. *Theorem 8.4 holds for arbitrary games.*

Note that Theorem 8.4 is a generalization of Monderer and Tennenholtz' Corollary 8.2 from [81]. They discovered that for a singleton x, it holds that x has a 0-implementation if and only if x is a Nash equilibrium. While their observation covers the special case of singleton-regions, our theorem holds for any strategy profile region. Unfortunately, for general regions, one direction of the equivalence holding for singletons does not hold anymore due to the fact that 0-implementable regions O must contain a player's best response to any o_{-i} but they need not contain best responses exclusively.

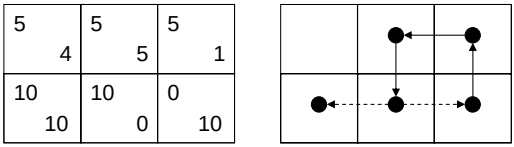


Figure 8.1: Sample game G with best response graph $\mathcal{G}_G$. The Nash equilibrium in the bottom left corner has no outgoing edges. The dotted arrows do not belong to the edge set of $\mathcal{G}_G$ as the row has multiple best responses.

8.3 Worst-Case Implementation Cost

We begin by studying exact implementations where the mechanism designer aims at implementing an *entire* strategy profile region based on her creditability. Subsequently, we examine general k-implementations.

Exact Implementation

Recall that the matrix V is an exact k-implementation of a strategy region O iff $X^*(V) = O$ and $\sum_{i=1}^{n} V_i(x) \leq k \ \forall x \in X^*(V)$, i.e. each strategy O_i is part of the set of player i's non-dominated strategies for all Players i. We present the first correct algorithm to find such implementations. Then we show that a bankrupt mechanism designer can determine in polynomial time whether a given region is implementable at zero cost. We will also establish an interesting connection between zero cost implementations and best response graphs.

Algorithm and Complexity

Recall that in our model each player classifies the strategies available to her as either dominated or non-dominated. Thereby, each dominated strategy

$x_i \in X_i \backslash X_i^*$ is dominated by at least one non-dominated strategy $x_i^* \in X_i^*$. In other words, a game determines for each player i a relation M_i^G from dominated to non-dominated strategies $M_i^G : X_i \backslash X_i^* \to X_i^*$, where $M_i^G(x_i) = x_i^*$ states that $x_i \in X_i \backslash X_i^*$ is dominated by $x_i^* \in X_i^*$. See Figure 8.2 for an example.

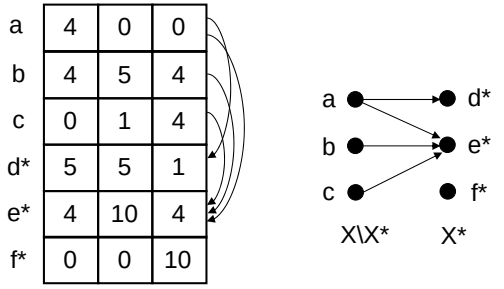


Figure 8.2: Game from a single player's point of view with the corresponding relation of dominated ($X_i \backslash X_i^* = \{a, b, c\}$) to non-dominated strategies ($X_i^* = \{d^*, e^*, f^*\}$).

When implementing a strategy profile region O exactly, the mechanism designer creates a modified game $G(V)$ with a new relation $M_i^V : X_i \setminus O_i \to O_i$ such that all strategies outside O_i map to at least one strategy in O_i. Therewith, the set of all newly non-dominated strategies of player i must constitute O_i. As every $V \in \mathcal{V}^*(O)$ determines a set of relations $M^V := \{M_i^V : i \in N\}$, there must be a set M^V for every V implementing O optimally as well. If we are given such an optimal relation set M^V without the corresponding optimal exact implementation, we can compute a V with minimal payments and the same relation M^V, i.e., given an optimal relation we can find an optimal exact implementation. As an illustrating example, assume an optimal relation set for G with $M_i^G(x_{i1}^*) = o_i$ and $M_i^G(x_{i2}^*) = o_i$. Thus, we can compute V such that o_i must dominate x_{i1}^* and x_{i2}^* in $G(V)$, namely, the condition $U_i(o_i, o_{-i}) + V_i(o_i, o_{-i}) \geq \max_{s \in (x_{i1}^*, x_{i2}^*)}(U_i(s, o_{-i}) + V_i(s, o_{-i}))$ must hold $\forall o_{-i} \in O_{-i}$. In an optimal implementation, Player i is not offered payments for strategy profiles of the form $(\bar{o}_i, x_{-i})$ where $\bar{o}_i \in X_i \backslash O_i,\ x_{-i} \in X_{-i}$. Hence, the condition above can be simplified to $V_i(o_i, o_{-i}) = \max(0, \max_{s \in \{x_{i1}^*, x_{i2}^*\}}(U_i(s, o_{-i}))) - U_i(o_i, o_{-i})$. Let $S_i(o_i) := \{s \in X_i \backslash O_i | M_i^V(s) = o_i\}$ be the set of strategies where M^V corresponds to an optimal exact implementation of O. Then, an implementation V with $V_i(\bar{o}_i, x_{-i}) = 0$, $V_i(o_i, \bar{o}_{-i}) = \infty$ for any player i, and $V_i(o_i, o_{-i}) = \max\left\{0, \max_{s \in S_i(o_i)}(U_i(s, o_{-i}))\right\} - U_i(o_i, o_{-i})$ is an optimal ex-

act implementation of O as well. Therefore, the problem of finding an optimal exact implementation V of O corresponds to the problem of finding an optimal set of relations $M_i^V : X_i \backslash O_i \to O_i$.

Our algorithm $\mathcal{ALG}_{exact}$ (cf. Algorithm 8.2) exploits this fact and constructs an implementation V for all possible relation sets, checks the cost that V would entail and returns the lowest cost found.

Algorithm 8.2 Exact k-Implementation ($\mathcal{ALG}_{exact}$)

Input: Game G, convex region O with $O_{-i} \subset X_{-i} \forall\ i$
Output: $k^*(O)$
1: $V_i(x) := 0$, $W_i(x) := 0\ \forall x \in X\ ,\ i \in N$;
2: $V_i(o_i, \bar{o}_{-i}) := \infty\ \forall i \in N,\ o_i \in O_i\ ,\ \bar{o}_{-i} \in X_{-i} \backslash O_{-i}$;
3: compute X^*;
4: **return** ExactK(V, n);

***ExactK(V, i)*:**
Input: payments V, current player i
Output: $k^*(O)$ for $G(V)$
1: **if** $\left|X_i^*(V) \backslash O_i\right| > 0$ **then**
2: $s :=$ any strategy in $X_i^*(V) \backslash O_i$; $k_{best} := \infty$;
3: **for all** $o_i \in O_i$ **do**
4: **for all** $o_{-i} \in O_{-i}$ **do**
5: $W_i(o_i, o_{-i}) := \max(0, U_i(s, o_{-i}) - (U_i(o_i, o_{-i}) + V_i(o_i, o_{-i})))$;
6: **od**
7: $k :=$ ExactK($V + W$, i);
8: **if** $k < k_{best}$ **then**
9: $k_{best} := k$;
10: **fi**
11: **for all** $o_{-i} \in O_{-i}$ **do**
12: $W_i(o_i, o_{-i}) := 0$;
13: **od**
14: **od**
15: **return** k_{best};
16: **else if** $i > 1$ **then**
17: **return** $ExactK(V, i-1)$;
18: **else**
19: **return** $\max_{o \in O} \sum_i V_i(o)$;
20: **fi**

Theorem 8.6. *$\mathcal{ALG}_{exact}$ computes a strategy profile region's optimal exact implementation cost in time*

$$\mathrm{O}\left(|X|^2 \max_{i \in N}(|O_i|^{n|X_i^* \backslash O_i|-1}) + n|O| \max_{i \in N}(|O_i|^{n|X_i^* \backslash O_i|})\right).$$

Proof. $\mathcal{ALG}_{exact}$ is correct as it checks all possible relations in the relation set $M^V = \{M_i^V : X_i^*(V) \backslash O_i \to O_i\ \forall i \in N\}$ recursively by calling the subroutine $ExactK$ in Line 6. Therefore, it must find the relation set which corresponds to an implementation with optimal cost.

It remains to prove the algorithm's runtime. Computing the non-dominated region X^* by checking for each strategy whether it is dominated takes time $\sum_{i=1}^{n} \binom{|X_i|}{2}|X_{-i}| = O(n|X|^2)$. The complexity of this computation asymptotically dominates the runtime required by Lines 1 and 2. We next examine the complexity of subroutine *ExactK*. Computing Line 1 costs $|X|^2$, the two for-loops in Lines 3 and 4 are executed $|O|$ times, and *ExactK* is called $|O_i|$ times (Line 6). Hence, we derive the following (asymptotic) recursive equations for the runtime $T_i(\ell)$ for $ExactK(V, i)$ if i has yet ℓ strategies to dominate:

$$T_i(\ell) = \begin{cases} |X|^2 + |O| + |O_i|T_i(\ell - 1) & \text{if } (0 < \ell < |X_i^* \setminus O_i|) \wedge (i \in N) \\ T_{i-1}(|X_{i-1}^* \setminus O_{i-1}|) & \text{if } \ell = 0 \wedge i \in N \\ n|O| & \text{if } \ell = 0 \wedge i = 0 \end{cases}$$

For $\ell_i = |X_i^* \setminus O_i|$, we obtain $T_i(\ell_i) = |O_i|^{\ell_i - 1}|X|^2 + |O_i|^{\ell_i} T_{i-1}(\ell_{i-1})$ if $i > 1$. Let $a_i = |O_i|^{\ell_i - 1}|X|^2, b_i = |O_i|^{\ell_i}$ and $a = \max_{i \in N} a_i, b = \max_{i \in N} b_i$; hence

$$\begin{aligned} T_i(\ell_i) &= a_i + b_i T_{i-1}(\ell_{i-1}) \\ &= a\left[\sum_{j=1}^{i}\prod_{k=1}^{j-1} b_k\right] + \left[\prod_{k=1}^{i} b_k\right] T_1(0) \\ &= a\sum_{j=1}^{i} b^{j-1} + b^i n|O| \\ &= ab^{i-1} + b^i n|O| \end{aligned}$$

and the claim follows. □

Note that $\mathcal{ALG}_{exact}$ has a large time complexity. In fact, a faster algorithm for this problem, called *Optimal Perturbation Algorithm* has been presented in [81]. In a nutshell, this algorithm proceeds as follows: After initializing V similarly to our algorithm, the values of the region O in the matrix V are increased slowly for every player i, i.e., by all possible differences between an agent's payoffs in the original game. The algorithm terminates as soon as all strategies in $X_i^* \setminus O_i$ are dominated. See 8.3 for a description in pseudocode.

Unfortunately, this algorithm does not always return an optimal implementation. In some cases, it increases the payments unnecessarily. The game depicted in Figure 8.3 is an example demonstrating that the optimal perturbation algorithm presented in [81] is not correct.

Algorithm 8.3 Perturbation Algorithm for two players [81]

Input: Game G, convex region O with $O_{-i} \subset X_{-i} \forall\, i$
Output: V satisfying $X^*(V) = O$

1: $M := \max_{i \in N,\ x \in X} U_i(x)$;
2: $E :=$ sorted set $\{e_l | e_l = U_i(x) - U_i(y),\ x, y \in X, i \in \{1,2\}\}$, i.e.,$(e_0, e_1, ..., e_k)$ is the list of possible differences between an agents' payoffs in U(the possible results one obtains by subtracting two possible payoffs of an agent in the given game G);
3: $V_i(x) := 0\ \forall x \in X,\ i \in \{1,2\}$;
4: $V_1(x_1, x_2) := M\ \forall x_1 \in O_1$ and $x_2 \in X_2 \setminus O_2$;
5: $V_2(x_1, x_2) := M\ \forall x_1 \in X_1 \setminus O_1$ and $x_2 \in O_2$;
6: $i := 0$;
7: **repeat**
8: $\quad V_1(o) := e_i\ \forall o \in O$;
9: $\quad i := i + 1$;
10: **until** $X_1^*(V) \equiv O_1$
11: $i := 0$;
12: **repeat**
13: $\quad V_2(o) := e_i\ \forall o \in O$;
14: $\quad i := i + 1$;
15: **until** $X_2^*(V) \equiv O_2$
16: **return** V;

G

2 0	0 0
0 0	2 3
4 0	0 0

V_{OPT}

2 0	5 0
0 3	5 0
0 5	0 0

$V_{PERTURB}$

2 3	5 0
2 3	5 0
0 5	0 0

Figure 8.3: Game G, X^* and O and payments V_{OPT}, $V_{PERTURB}$.

As can be verified easily, V_{OPT} implements O with cost $k = 3$. The matrix $V_{PERTURB}$ computed by the optimal perturbation algorithm implements O as well, however, it has cost $k = 5$. The set of possible differences between an agent's payoffs in the original game for G is $E = \{2, 3, 4\}$. We execute the steps 3 to 5 twice and obtain a perturbation game $G(V_M)$. After the steps 8 to 11 for $e_1 = 2$ we have generated $V(p_1, e_1)$ and $X^*(V')_1$ coincides with O_1. Executing steps 15 to 18 for player 2 three times until the condition in Line 15 is satisfied is responsible for the construction of $V(p_2, e_1)$ and $V(p_2, e_2)$. Thus, the perturbation algorithm returns the matrix $V_{PERTURB}$.

V_M

0 0	5 0
0 0	5 0
0 5	0 0

$V(p_1, e_1)$

2 0	5 0
2 0	5 0
0 5	0 0

$V(p_2, e_1)$

2 2	5 0
2 2	5 0
0 5	0 0

$V(p_2, e_2)$

2 3	5 0
2 3	5 0
0 5	0 0

Not only does this leave us without a polynomial algorithm, we even conjecture that the problem is inherently hard and that deciding whether an k-exact implementation exists is **NP**-hard.

Conjecture 1. Finding an optimal exact implementation of a strategy region is **NP**-hard.

Non-Exact Implementation

In contrast to exact implementations where the complete set of strategy profiles O must be non-dominated, the additional payments in non-exact implementations only have to ensure that a *subset* of O is the newly non-dominated region. Obviously, it matters which subset this is. Knowing that a subset $O' \subseteq O$ bears optimal cost, we could find $k(O)$ by computing $k^*(O')$. Apart from the fact that finding an optimal implementation includes solving the – believed to be **NP**-hard – optimal exact implementation cost problem for at least one subregion of O, finding this subregion might also be **NP**-hard since there are exponentially many possible subregions. In fact, a reduction from the SAT problem is presented in [81]. The authors show how to construct a 2-person game in polynomial time given a CNF formula such that the game has a 2-implementation if and only if the formula has a satisfying assignment. However, their proof is not correct: While there indeed exists a 2-implementation for every satisfiable formula, it can be shown that 2-implementations also exist for non-satisfiable formulas. E.g., strategy profiles $(x_i, x_i) \in O$ are always 1-implementable. Unfortunately, we were not able to correct their proof. However, we conjecture the problem to be **NP**-hard, i.e., we assume that no algorithm can do much better than performing a brute force computation of the exact implementation cost (cf. Algorithm 8.2) of all possible subsets, unless **NP** = **P**.

Conjecture 2. Finding an optimal implementation of a strategy region is **NP**-hard.

For the special case of zero cost regions, Theorem 8.4 implies the following.

Corollary 8.7. *If a strategy profile region O has zero implementation cost then the corresponding subgraph $\mathcal{G}_{G,O}$ in the game's best response graph contains a subgraph $\mathcal{G}_{G,O'}, O' \subseteq O$, with no outgoing edges.*

Corollary 8.7 is useful to a bankrupt mechanism designer since searching the game's best response graph for subgraphs without outgoing edges helps her spot candidates for regions which can be implemented by mere creditability. In general though, the fact that finding optimal implementations seems computationally hard raises the question whether there are polynomial time algorithms achieving good approximations. As mentioned

in Section 8.3, each V implementing a region O defines a domination relation $M_i^V : X_i \setminus O_i \to O_i$. This observation leads to the idea of designing heuristic algorithms that find a correct implementation by establishing a corresponding relation set $\{M_1, M_2, \ldots, M_n\}, M_i : X_i^* \setminus O_i \to O_i$ where each $x_i^* \in X_i^* \setminus O_i$ maps to at least one $o_i \in O_i$. These algorithms are guaranteed to find a correct implementation of O, however, the corresponding implementations may not be cost-optimal.

Our greedy algorithm $\mathcal{ALG}_{greedy}$ (cf. Algorithm 8.4) associates each strategy x_i^* yet to be dominated with the o_i with minimal distance Δ_G to x_i^*, i.e., the maximum value that has to be added to $U_i(x_i', x_{-i})$ such that x_i' dominates x_i: $\Delta_G(x_i, x_i') := \max_{x_{-i} \in X_{-i}} \max(0, U_i(x_i, x_{-i}) - U_i(x_i', x_{-i}))$. Similarly to the greedy approximation algorithm for the *set cover problem* [60, 78] which chooses in each step the subset covering the most elements not covered already, $\mathcal{ALG}_{greedy}$ selects a pair of (x_i^*, o_i) such that by dominating x_i^* with o_i, the number of strategies in $X_i^* \setminus O_i$ that will be dominated therewith is maximal. Thus, in each step there will be an o_i assigned to dominate x_i^* which has minimal dominating cost. Additionally, $\mathcal{ALG}_{greedy}$ takes any opportunity to dominate multiple strategies. $\mathcal{ALG}_{greedy}$ is described in detail in Algorithm 8.4. It returns an implementation V of O; to determine V's cost, one needs to compute $\max_{x^* \in X^*(V)} \sum_{i \in N} V_i(x^*)$.

Theorem 8.8. *$\mathcal{ALG}_{greedy}$ returns an implementation of a convex strategy profile region $O \in X$ in time*

$$\mathrm{O}\left(n|X|^2 \max_{i \in N} |X_i^* \setminus O_i| + n|X| \max_{i \in N} |X_i^* \setminus O_i|^3 \right).$$

Proof. $\mathcal{ALG}_{greedy}$ terminates since in every iteration of the while-loop, there is at least one newly dominated strategy. The payment matrix V returned is an implementation of O because the while-condition $X_i^*(V) \not\subseteq O_i$ turned false for all $i \in N$ and thus, it holds that $X_i^*(V) \subseteq O_i \ \forall i \in N$.

Line 1 takes time $\mathrm{O}(|X|\, n)$. Asymptotically, computing X^* costs $\mathrm{O}\big(|X|^2 n\big)$. Setting the payments $V_i(o_i, \bar{o}_{-i})$ to infinity (Line 4) for all players takes time $\mathrm{O}(n\,|X \setminus O|)$. The while-loop (Lines 5-18) is executed $|X_i^* \setminus O_i|$ times, and evaluating the while-loop's condition takes at most time $|X^2|$. One iteration of the while-loop takes time

$$\underbrace{|X_i^* \setminus O_i|}_{Line\ 7} \cdot \Big(\underbrace{|O_i||X_{-i}|}_{Line\ 8} + \underbrace{|O_{-i}|}_{Line\ 9} + \underbrace{|X_i^* \setminus O_i|}_{Line\ 12} \underbrace{(|X| + |O_{-i}|)}_{Line\ 13} \Big) + \underbrace{O_{-i}}_{Line 17}$$

Combining the above expressions yields the claim. □

$\mathcal{ALG}_{red}$ (cf. Algorithm 8.5) is a more sophisticated algorithm applying $\mathcal{ALG}_{greedy}$. Instead of terminating when the payment matrix V implements O, this algorithm continues to search for a payment matrix inducing even less cost. It uses $\mathcal{ALG}_{greedy}$ to approximate the cost repeatedly, varying the

region to be implemented. As $\mathcal{ALG}_{greedy}$ leaves the while-loop if $X_i^*(V) \subseteq O_i$, it might miss out on cheap implementations where $X_i^*(V) \subseteq Q_i$, $Q_i \subset O_i$. $\mathcal{ALG}_{red}$ examines some of these subsets as well by calling $\mathcal{ALG}_{greedy}$ for some Q_i. If we manage to reduce the cost, we continue with $O_i := Q_i$. We stop when neither the cost can be reduced anymore nor any strategies can be deleted from any O_i.

Theorem 8.9. *Let T_g denote the runtime of $\mathcal{ALG}_{greedy}$. $\mathcal{ALG}_{red}$ returns an implementation of O in time*

$$\mathrm{O}\left(n|O|\max_{i\in N}|O_i|(|O|+n+T_g)\right).$$

Proof. $\mathcal{ALG}_{red}$ terminates because the condition of the while-loop does not hold anymore if there are no more strategies left, and because in at most every $\max_{i\in N}|O_i|^{th}$ iteration at least one strategy is removed. Clearly, the while-loop is repeated at most $\max_{i\in N}|O_i|\cdot|O|$ times, as a removed strategy is never added again. We iterate over all players (time n) and look for a strategy to be removed (time $|O|$ in Line 5). Evaluating the if-clause (Line 6) requires time $n+|O_i|$. Finally in Line 10, the greedy algorithm is called recursively, which is assumed to take time T_g. The time complexity of all other operations can be neglected. □

An alternative heuristic algorithm for computing a region O's implementation cost retrieves the region's cheapest singleton, i.e., $\min_{o\in O} k(o)$, where a singleton's implementation cost is $k(o) = \min_{o\in O}\sum_{i\in N}\max_{x_i\in X_i}(U_i(x_i,o_{-i})-U_i(o_i,o_{-i}))$ [81]. The best singleton heuristic algorithm performs quite well for randomly generated games as our simulations reveal (cf. Section 8.3), but it can result in an arbitrarily large k in the worst case: Figure 8.4 depicts a game where each singleton o in the region O consisting of the four bottom left profiles has cost $k(o) = 11$ whereas V implements O at cost 2.

This raises the following question: What characteristics are stringent for a game and a corresponding desired strategy profile region O such that only non-singleton subregions bear the optimal implementation cost? Clearly, we have to consider games where at least one player has four or more strategies, at least two of which must not be in O_i. Moreover, it must cost less to dominate them with two strategies in O_i than with just one strategy in O_i.

Simulation

All our algorithms return correct implementations of the desired strategy profile sets and – apart from the recursive algorithm $\mathcal{ALG}_{exact}$ for the optimal exact implementation – run in polynomial time. In order to study the quality of the resulting implementations, we performed several simulations

Algorithm 8.4 Greedy Algorithm $\mathcal{ALG}_{greedy}$

Input: Game G, convex target region O
Output: Implementation V of O

```
1: V_i(x) := 0; W_i(x) := 0 ∀x ∈ X , i ∈ N;
2: compute X*;
3: for all i ∈ N do
4:    V_i(o_i, ō_{-i}) := ∞ ∀o_i ∈ O_i , ō_{-i} ∈ X_{-i}\O_{-i};
5:    while X*_i(V) ⊈ O_i do
6:       c_best := 0; m_best :=null; s_best :=null;
7:       for all s ∈ X*_i(V)\O_i do
8:          m := arg min_{o_i ∈ O_i}(Δ_{G(V)}(s, o_i));
9:          for all o_{-i} ∈ O_{-i} do
10:            W_i(m, o_{-i}):=max(0, U_i(s, o_{-i})− (U_i(m, o_{-i}) + V_i(m, o_{-i})));
11:         od
12:         c := 0;
13:         for all x ∈ X*_i \ O_i do
14:            if m dominates x in G(V + W) then
15:               c + +;
16:            fi
17:         od
18:         if c > c_best then
19:            c_best := c ; m_best := m ; s_best := s;
20:         fi
21:      od
22:      for all o_{-i} ∈ O_{-i} do
23:         V_i(m_best, o_{-i})+=max(0, U_i(s_best, o_{-i})−
            (U_i(m_best, o_{-i}) + V_i(m_best, o_{-i})));
24:      od
25:   od
26: od
27: return V;
```

comparing the implementation cost computed by the different algorithms. We have focused on two-person games using random game tables where both players have payoffs chosen uniformly at random from the interval $[0, max]$, for some constant max. We have also studied generalized *scissors, rock, paper games* (a.k.a., *Jan Ken Pon games*), that is, *symmetric zero-sum games* with payoff values chosen uniformly at random from an interval $[0, max]$. We find that – for the same interval and the same number of strategies – the average implementation cost of random symmetric zero-sum games, random symmetric games, and completely random games hardly deviate. This is probably due to the fact that in all examined types of random games virtually all strategies are non-dominated. Therefore, in the following, we present our results on symmetric random games only.

Algorithm 8.5 Reduction Algorithm $\mathcal{ALG}_{red}$

Input: Game G, convex target region O
Output: Implementation V of O

```
1:  [k, V] := greedy(G, O);
2:  k_temp := −1; c_i := ⊥ ∀i; T_i := {} ∀i;
3:  while (k > 0) ∧ (∃i : |O_i| > 1) ∧ (∃i : O_i ⊈ T_i) do
4:    for all i ∈ N do
5:      x_i := arg min_{o_i ∈ O_i} (max_{o_{−i} ∈ O_{−i}} U_i(o_i, o_{−i}));
6:      if (O_i ⊈ T_i) ∧ (¬c_j ∀j) ∧ (x_i ∈ T_i) then
7:        x_i := arg min_{o_i ∈ O_i \ {x_i}} (max_{o_{−i} ∈ O_{−i}} (U_i(o_i, o_{−i})));
8:      fi
9:      if |O_i| > 1 then
10:       O_i := O_i \ {x_i};
11:     fi
12:     [k_temp, V] := greedy(G, O);
13:     if k_temp ≥ k then
14:       O_i := O_i ∪ {x_i}; T_i := T_i ∪ {x_i}; c_i := ⊥;
15:     else
16:       k := k_temp; T_i := {} ∀i; c_i := ⊤;
17:     fi
18:   od
19: od
20: return V;
```

$G =$

20 / 0	11 / 9	15 / 15	15 / 15
11 / 9	20 / 0	15 / 15	15 / 15
19 / 10	10 / 19	9 / 11	0 / 20
10 / 19	19 / 10	0 / 20	9 / 11

$V =$

0 / ∞	0 / ∞	0 / 0	0 / 0
0 / ∞	0 / ∞	0 / 0	0 / 0
1 / 1	1 / 1	∞ / 0	∞ / 0
1 / 1	1 / 1	∞ / 0	∞ / 0

Figure 8.4: 2-player game where O 's optimal implementation V yields a region $|X^*(V)| > 1$.

Non-Exact Implementation

We observe that implementing the best singleton often yields low cost. In other words, especially when large sets have to be implemented, our greedy

algorithms tend to implement too many strategy profiles and consequently incur unnecessarily high cost. However, while this is often true in random games, there are counter examples where the cheapest singleton is costly compared to the implementation found by the greedy algorithms; Figure 8.4 depicts a situation where the greedy algorithm computes a better solution. We presume that $\mathcal{ALG}_{red}$ might improve relatively to the best singleton heuristic algorithm for larger player sets.

Figure 8.5 plots the implementation cost determined by the $\mathcal{ALG}_{greedy}$, $\mathcal{ALG}_{red}$ and the singleton algorithm as a function of the number of strategies involved. On average, the singleton algorithm performed much better than the other two, with $\mathcal{ALG}_{greedy}$ being the worst of the candidates. In the second plot we can see the implementation cost the algorithms compute for different payoff value intervals $[0, max]$. As expected, the implementation cost increases for larger intervals.

Exact Implementation

The observation that the greedy algorithm $\mathcal{ALG}_{greedy}$ implements rather large subregions of O suggests that it may achieve good results for exact implementations. We can modify an implementation V of O, which yields a subset of O, without changing any entry $V_i(o), o \in O$, such that the resulting V implements O *exactly*.

Theorem 8.10. *If $O_{-i} \subset X_{-i}\ \forall i \in N$, it holds that $k^*(O) \leq \max_{o \in O} V(o)$ for an implementation V of O.*

Proof. If V is a non-exact implementation of O, there are some strategies O_i dominated by other strategies in O_i. A dominated strategy o_i can be made non-dominated by adding payments to the existing V_i for profiles of the form $(o_i, \bar{o}_{-i})$, where $\bar{o}_{-i} \in X_{-i} \backslash O_{-i}$. Let $a \in O_i$ dominate $b \in O_i$ in $G(V)$. The interested party can annihilate this relation of a dominating b by choosing payment $V_i(b, \bar{o}_{-i})$ such that Player i's resulting payoff $U_i(b, \bar{o}_{-i}) + V_i(b, \bar{o}_{-i})$ is larger than $U_i(a, \bar{o}_{-i}) + V_i(a, \bar{o}_{-i})$ and therefore a does not dominate b anymore. As such, all dominations inside O_i can be neutralized even if $|X_{-i} \setminus O_{-i}| = 1$. One must realize that the relation of domination is *irreflexive* and *transitive* and therefore establishes a strict order among the strategies. Let $\bar{o}_{-i} \in X_{-i} \backslash O_{-i}$ be a column in player i's payoff matrix outside O. By choosing the payments $V_i(o_i, \bar{o}_{-i})$ such that the resulting payoffs $U_i(o_i, \bar{o}_{-i}) + V_i(o_i, \bar{o}_{-i})$ establish the same order with the *less-than relation* $(<)$ as the strategies o_i with the domination relation, all $o_i \in O_i$ will be non-dominated. Thus, a V' can be constructed from V which implements O exactly without modifying any entry $V_i(o), o \in O$. □

Theorem 8.10 enables us to use $\mathcal{ALG}_{greedy}$ for an exact cost approximation by simply computing $\max_{o \in O} V(o)$ instead of $\max_{x \in X^*(V)} V(x)$. Fig-

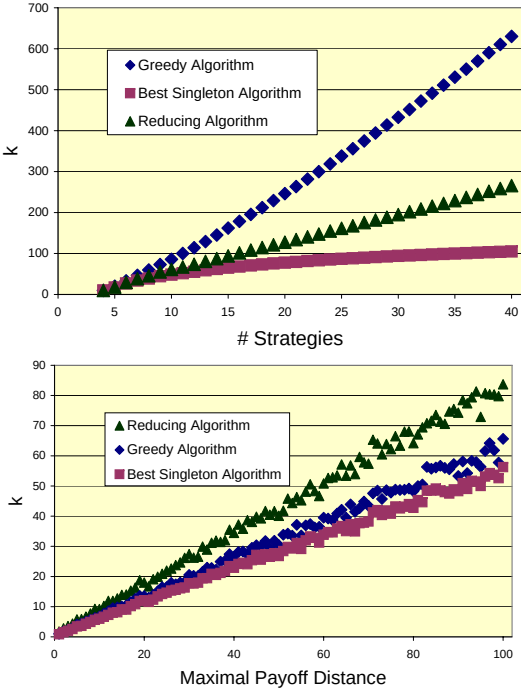


Figure 8.5: The average implementation cost k of sets O over 100 random games where $|O_i| = \lfloor n/3 \rfloor$. *Left*: utility values chosen uniformly at random from $[0, 20]$. For different intervals we obtain approximately the same result when normalizing k with the maximal possible value. *Right*: eight strategies are used; other numbers of strategies yield similar results.

ure 8.6 depicts the exact implementation cost determined by $\mathcal{ALG}_{greedy}$, $\mathcal{ALG}_{exact}$ and the perturbation algorithm from [81]. The first figure plots k as a function of the number of strategies, whereas the second figure demonstrates the effects of varying the size of the payoff interval. Due to the large runtime of $\mathcal{ALG}_{exact}$, we were only able to compute k for a small number of strategies. However, for these cases, our simulations reveal that $\mathcal{ALG}_{greedy}$ often finds implementations which are close to optimal and that it is better than the perturbation algorithm. For different payoff value intervals $[0, max]$, we observe a faster increase in k than in the non-exact implementation case. This suggests that implementing a smaller region entails lower cost for random games on average.

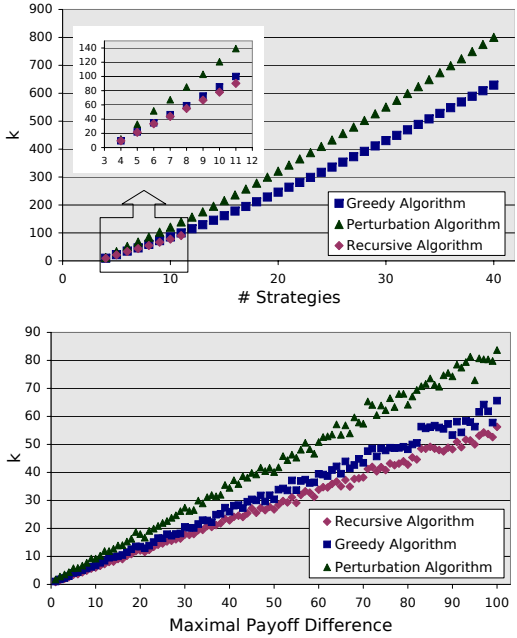


Figure 8.6: The average exact implementation cost k of sets O over 100 random games where $|O_i| = \lfloor n/3 \rfloor$. *Left*: utility values chosen uniformly at random from $[0, 20]$. For other intervals we obtain approximately the same result when normalizing k with the maximal possible value. *Right*: eight strategies are used; the plot is similar for other numbers of strategies.

Furthermore, our simulations revealed that the variance of the cost found decreases with the number of strategies for all algorithms, while it remains roughly constant for intervals of various size. The variance of the singleton heuristic is typically smaller than the variance of $\mathcal{ALG}_{greedy}$ and $\mathcal{ALG}_{red}$. The same holds for exact implementations, where the perturbation algorithm has the largest variance.

Finally, we tested different options to choose the next strategy in $\mathcal{ALG}_{greedy}$ (Line 8) and $\mathcal{ALG}_{red}$ (Lines 5 and 7). However, none of the alternatives we tested performed better than the ones described in Section 2.3.

In conclusion, our simulations have shown that for the case of non-exact implementations, there are interesting differences between the algorithms proposed in Section 2.3. In particular, the additional reductions by $\mathcal{ALG}_{red}$ are beneficial. For the case of exact implementations, our modified greedy

algorithm yields good results. As a final remark we want to mention that, although $\mathcal{ALG}_{greedy}$ and $\mathcal{ALG}_{red}$ may find cheap implementations in the average case, there are examples where the approximation ratio of these algorithms is large.

8.4 Uniform Implementation Cost

We will now turn our attention to the situation of less anxious mechanism designers who anticipate uniform rather than worst-case implementation cost. They assume the players to select one of their non-dominated strategies uniformly at random. This is a reasonable assumption if the players do only know their own utilities.

Complexity

In the following we show that it is **NP**-hard to compute the uniform implementation cost for both the non-exact and the exact case. We devise game configurations which reduce SET COVER to the problem of finding an implementation of a strategy profile set with optimal uniform cost.

Theorem 8.11. *In games with at least two (three) players, the problem of finding a strategy profile set's exact (non-exact) uniform implementation cost is* **NP***-hard.*

Proof. Exact Case: For a given universe $\mathcal{U}$ of l elements $\{e_1, e_2, \ldots, e_l\}$ and m subsets $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$, with $S_i \subset \mathcal{U}$, SET COVER is the problem of finding the minimal collection of S_i's which contains each element $e_i \in \mathcal{U}$. We assume without loss of generality that $\nexists (i \neq j) : S_i \subset S_j$. Given a SET COVER problem instance $SC = (\mathcal{U}, \mathcal{S})$, we can efficiently construct a game $G = (N, X, U)$ where $N = \{1, 2\}$, $X_1 = \{e_1, e_2, \ldots, e_l, s_1, s_2, \ldots, s_m\}$, and $X_2 = \{e_1, e_2, \ldots, e_l, d, r\}$. Each strategy e_j corresponds to an element $e_j \in \mathcal{U}$, and each strategy s_j corresponds to a set S_j. Player 1's payoff function U_1 is defined as follows: $U_1(e_i, e_j) := m + 1$ if $i = j$ and $U_1(e_i, e_j) := 0$ otherwise, $U_1(s_i, e_j) := m + 1$ if $e_j \in S_i$ and $U_1(s_i, e_j) := 0$ otherwise, $U_1(e_i, d) := 1$, $U_1(s_i, d) := 0$, $U_1(x_1, r) := 0$ $\forall x_1 \in X_1$. Player 2 has a payoff of 0 when playing r and 1 otherwise. In this game, strategies e_j are not dominated for player 1 because in column d, $U_1(e_j, d) > U_1(s_i, d)$, $\forall i \in \{1, \ldots m\}$. The set O we would like to implement is $\{(x_1, x_2) | x_1 = s_i \wedge (x_2 = e_i \vee x_2 = d)\}$. See Figure 3 for an example. Let $Q = \{Q_1, Q_2, \ldots, Q_k\}$, where each Q_j corresponds to an S_i. We now claim that Q is an optimal solution for a SET COVER problem, an optimal exact implementation V of O in the corresponding game has payments $V_1(s_i, d) := 1$ if $Q_i \in Q$ and $V_1(s_i, d) := 0$ otherwise, and all payments $V_1(s_i, e_j) := 0$.

Note that by setting $V_1(s_i, d)$ to 1, strategy s_i dominates all strategies e_i which correspond to an element in S_i. Thus, our payment matrix makes all

	e_1	e_2	e_3	e_4	e_5	d	r
e_1	5	0	0	0	0	1	0
e_2	0	5	0	0	0	1	0
e_3	0	0	5	0	0	1	0
e_4	0	0	0	5	0	1	0
e_5	0	0	0	0	5	1	0
s_1	5	0	0	5	0	0	0
s_2	0	5	0	5	0	0	0
s_3	0	5	5	0	5	0	0
s_4	5	5	5	0	0	0	0

Figure 8.7: Payoff matrix for player 1 in a game which reduces the SET COVER problem instance $SC = (\mathcal{U}, \mathcal{S})$ where $\mathcal{U} = \{e_1, e_2, e_3, e_4, e_5\}$, $\mathcal{S} = \{S_1, S_2, S_3, S_4\}$, $S_1 = \{e_1, e_4\}, S_2 = \{e_2, e_4\}, S_3 = \{e_2, e_3, e_5\}, S_4 = \{e_1, e_2, e_3\}$ to the problem of computing $k^*_{UNI}(O)$. The optimal exact implementation V of O in this sample game adds a payment V_1 of 1 to the strategy profiles (s_1, d) and (s_3, d), implying that the two sets S_1 and S_3 cover $\mathcal{U}$ optimally.

strategies e_i of player 1 dominated since any strategy e_i representing element e_i is dominated by the strategies s_j corresponding to S_j which cover e_i in the minimal covering set.[1] If there are any strategies s_i dominated by other strategies s_j, we can make them non-dominated by adjusting the payments $V_1(s_i, r)$ for column r. Hence, any solution of SC corresponds to a valid exact implementation of O.

It remains to show that such an implementation is indeed optimal and there are no other optimal implementations not corresponding to a minimal covering set. Note that by setting $V_1(s_i, d) := 1$ and $V_1(s_i, r) > 0$ for all s_i, all strategies e_j are guaranteed to be dominated and V implements O exactly with uniform cost $\varnothing_{o \in O} V(o) = m/|O|$. If an implementation had a positive payment for any strategy profile of the form (s_i, e_j), it would cost at least $m+1$ to have an effect. However, a positive payment greater than m yields a larger. Thus, an optimal V has positive payments inside set O only in column d. By setting $V_1(s_i, d)$ to 1, s_i dominates the strategies e_j which correspond to the elements in S_i, due to our construction. An optimal implementation

[1] If $|S_j| = 1$, s_j gives only equal payoffs in $G(V)$ to those of e_i in the range of O_2. However, s_j can be made dominating e_i by increasing s_j's payments $V_1(s_j, r)$ in the extra column r.

has a minimal number of 1s in column d. This can be achieved by selecting those rows s_i ($V_1(s_i, d) := 1$), which form a minimal covering set and as such all strategies e_i of player 1 are dominated at minimal cost. Our reduction can be generalized for $n > 2$ by simply adding players with only one strategy and zero payoffs in all strategy profiles.

Non-Exact Case: We give a similar reduction of SET COVER to the problem of computing $k_{UNI}(O)$ by extending the setup we used for proving the exact case. We add a third player and show **NP**-hardness for $n = 3$ first and indicate how the reduction can be adapted for games with $n > 3$. Given a SET COVER problem instance $SC = (\mathcal{U}, \mathcal{S})$, we can construct a game $G = (N, X, U)$ where $N = \{1, 2, 3\}$, $X_1 = \{e_1, e_2, \ldots, e_l, s_1, s_2, \ldots, s_m\}$, $X_2 = \{e_1, e_2, \ldots, e_l, s_1, s_2, \ldots, s_m, d, r\}$, $X_3 = \{a, b\}$. Again, each strategy e_j corresponds to an element $e_j \in \mathcal{U}$, and each strategy s_j corresponds to a set S_j. In the following, we use '_' in profile vectors as a placeholder for any possible strategy. Player 1's payoff function U_1 is defined as follows: $U_1(e_i, e_j, \_) := (m + l)^2$ if $i = j$ and 0 otherwise, $U_1(e_i, s_j, \_) := 0$, $U_1(s_i, e_j, \_) := (m + l)^2$ if $e_j \in S_i$ and 0 otherwise, $U_1(s_i, s_j, \_) := 0$ if $i = j$ and $(m + l)^2$ otherwise, $U_1(e_i, d, \_) := 1$, $U_1(s_i, d, \_) := 0$, $U_1(\_, r, \_) := 0$. Player 2 has a payoff of $(m+l)^2$ for any strategy profile of the form $(s_i, s_i, \_)$ and 0 for any other strategy profile. Player 3 has a payoff of $m + l + 2$ for strategy profiles of the form (s_i, s_i, b), a payoff of 2 for profiles (s_i, e_i, b) and profiles $(s_i, s_j, b), i \neq j$, and a payoff of 0 for any other profile. The set O we would like to implement is $\{(x_1, x_2, x_3) | x_1 = s_i \wedge (x_2 = e_i \vee x_2 = s_i \vee x_2 = d) \wedge (x_3 = a)\}$. See Figure 8.8 for an example.

First, note that any implementation of O will have $V_3(o_1, o_2, a) \geq U_3(o_1, o_2, b)$, in order to leave player 3 no advantage when playing b instead of a. In fact, setting $V_3(o_1, o_2, a) = U_3(o_1, o_2, b)$ suffices.[2] Also note that for player 2, O_2 can be made non-dominated without offering any payments inside O, e.g., set $V_2(e_i, e_j, \_) = 1$ and $V_2(e_i, d, \_) = 1$.

Analogously to the exact case's proof, we claim that if and only if Q=$\{Q_1, Q_2, \ldots, Q_k\}$, where each Q_j corresponds to an S_i, is an optimal solution for a SET COVER problem, there exists an optimal exact implementation V of O in the corresponding game. This implementation selects a row s_i ($V_1(s_i, d, a) = 1$), if $Q_i \in Q$ and does not select s_i ($V_1(s_i, d, a) = 0$) otherwise. All other payments V_1 inside O are 0. Player 2's payments $V_2(o)$ are 0 for all $o \in O$ and player 3's payoffs are set to $V_3(o_1, o_2, a) = U_3(o_1, o_2, b)$. A selected row s_i contributes $cost_{s_i} = (3(l + m) + 1)/(l + m + 1)$. A non-selected row s_j contributes $cost_{s_j} = (3(l + m))/(l + m + 1) < cost_{s_i}$. Thus including non-selected rows in $X^*(V)$ can be profitable. Selecting all rows s_i yields a correct implementation of O with uniform cost $\varnothing_{i=1}^{m} cost_{s_i} = (3(l + m) + 1)/(l + m + 1) < 3$.

[2] Setting any $V_3(a, \bar{o}_{-3}) > U_3(b, \bar{o}_{-3})$ where $\bar{o}_{-3}$ is outside O lets player 3 choose strategy a.

	e_1	e_2	e_3	e_4	e_5	s_1	s_2	s_3	s_4	d	r
e_1	81 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	1 0	0 0
e_2	0 0	81 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	1 0	0 0
e_3	0 0	0 0	81 0	0 0	0 0	0 0	0 0	0 0	0 0	1 0	0 0
e_4	0 0	0 0	0 0	81 0	0 0	0 0	0 0	0 0	0 0	1 0	0 0
e_5	0 0	0 0	0 0	0 0	81 0	0 0	0 0	0 0	0 0	1 0	0 0
s_1	81 0	0 0	0 0	81 0	0 0	0 81	81 0	81 0	81 0	0 0	0 0
s_2	0 0	81 0	0 0	81 0	0 0	81 0	0 81	81 0	81 0	0 0	0 0
s_3	0 0	81 0	81 0	0 0	81 0	81 0	81 0	0 81	81 0	0 0	0 0
s_4	81 0	81 0	81 0	0 0	0 0	81 0	81 0	81 0	0 81	0 0	0 0

	e_1	e_2	e_3	e_4	e_5	s_1	s_2	s_3	s_4	d	r
e_1	0	0	0	0	0	0	0	0	0	0	0
e_2	0	0	0	0	0	0	0	0	0	0	0
e_3	0	0	0	0	0	0	0	0	0	0	0
e_4	0	0	0	0	0	0	0	0	0	0	0
e_5	0	0	0	0	0	0	0	0	0	0	0
s_1	2	2	2	2	2	11	2	2	2	0	0
s_2	2	2	2	2	2	2	11	2	2	0	0
s_3	2	2	2	2	2	2	2	11	2	0	0
s_4	2	2	2	2	2	2	2	2	11	0	0

Figure 8.8: Payoff matrix for player 1 and Player 2 given Player 3 chooses a and payoff matrix for Player 3 when she plays strategy b in a game which reduces a SET COVER instance $SC = (\mathcal{U}, \mathcal{S})$ where $\mathcal{U} = \{e_1, e_2, e_3, e_4, e_5\}$, $\mathcal{S} = \{S_1, S_2, S_3, S_4\}$, $S_1 = \{e_1, e_4\}, S_2 = \{e_2, e_4\}, S_3 = \{e_2, e_3, e_5\}, S_4 = \{e_1, e_2, e_3\}$ to the problem of computing $k_{UNI}(O)$. Every implementation V of O in this game needs to add any positive payment in the second matrix to V_3, i.e. $V_3(x_1, x_2, a) = U_3(x_1, x_2, b)$, in order to convince player 3 of playing strategy a. An optimal implementation adds a payment V_1 of 1 to the strategy profiles (s_1, d, a) and (s_3, d, a), implying that the two sets S_1 and S_3 cover $\mathcal{U}$ optimally in the corresponding SET COVER problem.

In fact, the game's payoffs are chosen such that it is not worth implementing any set smaller than O. We show that for every set smaller than O the exact uniform implementation cost is strictly larger. Assume a set yielding lower cost implements α strategies for player 1, β strategies e_i and γ strategies s_j for player 2. Note that implementing player 2's strategy d is profitable if $\beta + \gamma > 0$, as it adds α to the denominator and at most α to the numerator of the implementation cost of sets without d. Consequently, there are three cases to consider: (i) $\beta \neq 0, \gamma = 0$: The costs add up to $\sum_{o \in O}(V_1(o) + V_2(o) + V_3(o))/|O| \geq (1 + (m+l)^2 + 2\alpha\beta)/(\alpha(\beta+1))$, which is greater than 3, since $\alpha \leq m, \beta \leq l$. (ii) $\beta = 0, \gamma \neq 0$: The aggregated cost is at least $(1 + \alpha(m+l) + 2\alpha\gamma)/(\alpha(\gamma+1))$, which is also greater than 3. (iii) $\beta \neq 0, \gamma \neq 0$: Assume there are κ sets necessary to cover U. Hence the sum of the payments in column d is at least κ. In this case, the cost amounts to $(\kappa + \alpha(m+l) + 2\alpha(\beta+\gamma))/(\alpha(\beta+\gamma+1)) = 2 + (m+l-2+\kappa/\alpha)/(\beta+\gamma+1) \geq k^*(O)$. Equality only holds if $\alpha = \gamma = m$ and $\beta = l$. We can conclude that O has to be implemented exactly in order to obtain minimal cost.

Therefore, an optimal implementation yields $X^*(V) = O$ with the inalienable payments to player 3 and a minimal number of 1-payments to player 1 for strategy profiles (s_i, d, a) such that every e_j is dominated by at least one s_i. The number of 1-payments is minimal if the selected rows correspond to a minimal covering set, and the claim follows.

Note that a similar SET COVER reduction can be found for games with more than three players. Simply add players to the described 3-player game with only one strategy. □

Due to the nature of the reduction the inapproximability results of SET COVER [5, 42] carry over to our problem.

Theorem 8.12. *No polynomial-time algorithm can achieve an approximation ratio better than* $\Omega\left(n \max_i\{\log |X_i^* \setminus O_i|\}\right)$ *for both the exact and non-exact implementation cost within any function of the input length unless* **P**=**NP**.

Proof. Exact Case: In order to prove this claim, a reduction similar to the one in the proof of Theorem 8.11 can be applied. Consider again a SET COVER instance with a universe $\mathcal{U}$ of l elements $\{e_1, e_2, \ldots, e_l\}$ and m subsets $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$, with $S_j \subset \mathcal{U}$. We construct a game $G = (N, X, U)$ with n players $N = \{1, \ldots, n\}$, where $X_i = \{e_1, e_2, \ldots, e_l, s_1, s_2, \ldots, s_m\}$ $\forall i \in \{1, \ldots, n-1\}$, and $X_n = \{e_1, e_2, \ldots, e_l, d, r\}$. Again, each strategy e_j corresponds to an element $e_j \in \mathcal{U}$, and each strategy s_j corresponds to a set S_j. Player i's payoff function U_i, for $i \in \{1, \ldots, n-1\}$, is defined as follows: Let e_k and s_k be strategies of player i and let e_l be a strategy of Player n. If $k = l$, player i has payoff $m+1$, and 0 otherwise. Moreover, $U_i(s_k, e_l) := m+1$ if $e_l \in S_k$ and 0 otherwise, and $U_i(e_k, d) := 1$, $U_i(s_k, d) := 0$, $U_i(x_k, r) := 0$ $\forall x_k \in X_i$. Thus, player i's payoffs only depend on Player i and Player n's

strategies. Player n has a payoff of 0 when playing r and 1 otherwise, independently of all other players' choices. We ask for an implementation of set O where Player i, for $i \in \{1, \ldots, n-1\}$, plays any strategy s_k, and Player n plays any strategy e_l or strategy d.

Due to the independence of the players' payoffs, the situation is similar to the example in Figure 3, and a SET COVER instance has to be solved for each player i $\forall i \in \{1, \ldots, n-1\}$. According to the well-known inapproximability results for SET COVER, no polynomial time algorithm exists which achieves a better approximation ratio than $\Omega(\log |X_i^* \setminus O_i|)$ for each player i, unless $\mathbf{P} = \mathbf{NP}$, and the claim follows.

Non-Exact Case: We use the inapproximability results for SET COVER again. Concretely, we assume a set of $n = 3k$ players for an arbitrary constant $k \in \mathbb{N}$ and make k groups of three players each. The payoffs of the three players in each group are the same as described in the proof of Theorem 8.11 for the non-exact case, independently of all other players' payoffs. Hence, SET COVER has to be solved for $n/3$ players. □

8.5 Alternative Rationality Models

Mechanism design by creditability offers many interesting extensions. In this section, two alternative models of rationality are introduced. If we assume that players do not just select *any* non-dominated strategy, but have other parameters influencing their decision process, our model has to be adjusted. In many (real world) games, players typically do not know which strategies the other players will choose. In this case, a player cannot do better than assume the other players to select a strategy *at random*. If a player wants to maximize her gain, she will take the *average payoff* of strategies into account. This kind of decision making is analyzed in the subsequent section. Afterwards, risk-averse players are examined. Finally, we take a brief look at the dynamics of repeated games with an interested third party offering payments *in each round*.

8.5.1 Average Payoff Model

As a player may choose any non-dominated strategy, it is reasonable to compute the payoff which each of her strategy will yield *on average*. Thus, assuming no knowledge on the payoffs of the other players, each strategy x_i has an average payoff of $p_i(x_i) := \frac{1}{|X_{-i}|} \sum_{x_{-i} \in X_{-i}} U_i(x_i, x_{-i})$ for player i. Player i will then select the strategy $s \in X_i$ with the largest $p_i(s)$, i.e., $s = \arg\max_{s \in X_i} p_i(s)$. If multiple strategies have the same average payoff, she plays one of them uniformly at random. For such average strategy games, we say that x_i *dominates* x_i' iff $p_i(x_i) > p_i(x_i')$. Note that with this modified

meaning of domination, the region of non-dominated strategies, X^*, differs as well.

The average payoff model has interesting properties, e.g., singleton profiles can be implemented for free.

Theorem 8.13. *If players maximize their average payoff, singleton strategy profiles are always 0-implementable if there are at least two players with at least two strategies.*

Proof. Let z be the strategy profile to be implemented. In order to make player i choose strategy z_i, the interested party may offer payments for any strategy profile $(z_i, \bar{z}_{-i})$ where $\bar{z}_{-i} \in X_{-i} \setminus \{z_{-i}\}$ such that $p_i(z_i)$ becomes player i's largest average payoff in $G(V)$. Since each player has at least two strategies to choose from, there is at least one $\bar{z}_{-i}$, and by making $V_i(z_i, \bar{z}_{-i})$ large enough (e.g., $V_i(z_i, \bar{z}_{-i}) := \max_{x_i \in X_i} \sum_{x_{-i} \in X_{-i}} U_i(x_i, x_{-i}) + \epsilon$) this can always be achieved. Therefore, z can be implemented without promising any payments for z. □

Theorem 8.13 implies that entire strategy profile regions O are 0-implementable as well: we just have to implement any singleton inside O.

Corollary 8.14. *In average strategy games where every player has at least two strategies, every strategy profile region can be implemented for free.*

Exact implementations can be implemented at no cost as well.

Theorem 8.15. *In average strategy games where $O_{-i} \subset X_{-i} \ \ \forall i \in N$, each strategy profile region has an exact 0-implementation.*

Proof. The mechanism designer can proceed as follows. First define $\mu_i := \max_{x_i \in X_i}\{p_i(x_i)\}$. Then set

$$V_i(o_i, \bar{o}_{-i}) := |X_{-i}|(\mu_i - p_i(x_i)) - U_i(x_i, x_{-i}) + \epsilon,$$

$\forall o_i \in O_i$, $\bar{o}_{-i} \in X_{-i} \setminus O_{-i}$. Consequently, it holds that for each player i and two strategies $x_i \in O_i$ and $x_i' \notin O_i$, $p_i(x_i) > p_i(x_i')$; moreover, no strategy $x_i \in O_i$ is dominated by any other strategy. As payments in $V_i(o_i, \bar{o}_{-i})$ with $o_i \in O_i$ and $\bar{o}_{-i} \in X_{-i} \setminus O_{-i}$ do not contribute to the implementation cost, Theorem 8.15 follows. □

8.5.2 Risk-Averse Players

Instead of striving for a high payoff on average, the players might be cautious or *risk-averse*. To account for such behavior, we adapt our model by assuming that the players seek to minimize the risk on missing out on benefits. In order to achieve this objective, they select strategies where the minimum gain is not less than any other strategy's minimum gain. If there is more than

one strategy with this property, the risk-averse player can choose a strategy among these, where the average of the benefits is maximal. More formally, let

$$min_i := \max_{x_i \in X_i} (\min_{x_{-i} \in X_{-i}} (U_i(x_i, x_{-i})))$$

and

$$\underset{X}{\varnothing} f(x) := \frac{1}{|X|} \cdot \sum_{x \in X} f(x).$$

Then player i selects a strategy m out of the strategy set

$$M_i = \{x_i | \forall x_{-i} \quad U_i(x_i, x_{-i}) = min_i\}$$

satisfying

$$m = \arg\max_{m \in M_i} (\underset{X_{-i}}{\varnothing} U_i(m, x_{-i})).$$

Theorem 8.16. *For risk-averse players the implementation cost of a singleton $z \in X$ is $k(z) = \sum_{i=1}^{n} \max(0, min_i - U_i(z))$.*

Proof. We show how to construct V implementing z with cost k and then prove that we cannot reduce the payments of $V(z)$. Since in this model every player i makes her decision without taking into account the benefits other players might or might not obtain, it suffices to consider each V_i separately. To ensure that player i selects z_i we have to set $V_i(z_i, x_{-i})$ to a value such that min_i is reached in $G(U + V)$ for each x_{-i}. Consequently we assign $V_i(z_i, x_{-i}) = \max(0, min_i - U_i(z_i, x_{-i}))$. We have to satisfy a second condition such that z_i is chosen, namely, $z_i = \arg\max_{m \in M}(\varnothing_{X_{-i}} U_i(m, x_{-i}))$. This is achieved by setting $V_i(z_i, x_{-i}) = \infty \quad \forall x_{-i} \neq z_{-i}$. We repeat these steps for all Players i. Clearly, V constructed in this manner implements z. Since the cost k only comprises the additional payments in $V(z)$ and lowering $V_i(z)$ for any i results in player i choosing a different strategy, we can deduce the statement of the theorem. □

For strategy profile regions, the situation with risk-averse players differs from the standard model considerably.

Theorem 8.17. *For risk-averse players the implementation cost for a strategy profile region $O \subset X$ is $k(O) = \min_{o \in O} \sum_{i=1}^{n} \max(0, min_i - U_i(o))$.*

Proof. Since we have to add up the cost to reach the required minimum for every strategy profile in $X^*(V)$ it cannot cost less to exactly implement more than one strategy profile, i.e., find V such that $|X^*(V)| = 1$. Thus V implementing the "cheapest" singleton in O yields an optimal implementation for O, and the claim follows. □

Algorithm 8.6 $\mathcal{ALG}_{risk}$Risk-averse Players: Exact Implementation

Input: Game G, target region O, $O_i \cap X_i^* = \emptyset \; \forall i \in N$
Output: V such that $X^*(V) = O$

1: compute X^*;
2: $V_i(z) = 0$ for all $i \in N, z \in X$;
3: **for all** $i \in N$ **do**
4: $V_i(x_i, x_{-i}) := \infty \quad \forall x_i \in O_i,\ x_{-i} \in X_{-i} \setminus O_{-i}$;
5: $V_i(x_i, x_{-i}) := \max(0, min_i - U_i(x_i, x_{-i})) \quad \forall x_i \in O_i,\ x_{-i} \in X_{-i}$;
6: **if** $O_{-i} = X_{-i}$ **then**
7: **if** $\tau(O_i) > \tau(X_i^*)$ **then**
8: **if** $|X_i| + \epsilon|O_i| > |X_i| + \sum_{o_i} \delta(o_i)$ **then**
9: $V_i(o_i, x_{-i}) := V_i(o_i, x_{-i}) + \delta(o_i) \quad \forall o_i, x_{-i}$;
10: **else**
11: $V_i(o_i, x_{-i}) := V_i(o_i, x_{-i}) + \epsilon \quad \forall o_i, x_{-i}$;
12: **fi**
13: **else**
14: **if** $\epsilon|O_i| > \sum_{o_i} [\epsilon + \delta(o_i)]$ **then**
15: $V_i(o_i, x_{-i}) := V_i(o_i, x_{-i}) + \epsilon + \delta(o_i) \quad \forall o_i, x_{-i}$;
16: **else**
17: $V_i(o_i, x_{-i}) := V_i(o_i, x_{-i}) + \epsilon \quad \forall o_i, x_{-i}$;
18: **fi**
19: **fi**
20: **fi**
21: **od**
22: **return** V;

Since it is always cheaper to implement a singleton than a region under this behavior model, there are no obvious reasons to implement a set of profiles *exactly*. Nonetheless, for completeness, we state the exact implementation cost for risk-averse players as well.

Theorem 8.18. *The problem of computing the exact implementation cost of a region is in* **P** *for risk-averse players.*

Proof. $\mathcal{ALG}_{risk}$ demonstrate how to compute $V_i(o)$ such that V implements the entire region O optimally. For a Player i and a set of strategies $Y_i \subseteq X_i$, we define

$$\tau(Y_i) := \max_{x_i \in Y_i} (\underset{X_{-i}}{\varnothing} ((U + V)_i(x_i, x_{-i})))$$

to be the maximum of the average benefits over all strategies. For each strategy of a Player i, we define $\delta(x_i) := \max(\tau(O_i), \tau(X_i^*)) - \varnothing_{X_{-i}}((U + V)_i(x_i, x_{-i})))$, for $x_i \in X_i$, to be the difference of the averages. Algorithm 8.6 constructs V if the target region O and X^* are disjoint. Analogously to the proofs above we can deal with each player i individually. The algorithm computes for all cases how much the interested party has to offer at least for strategy profiles in O and sets $V_i(x_i, x_{-i})$ to infinity for all $x_i \in O_i$, $x_{-i} \in X_{-i} \setminus O_{-i}$ (Line 4). Then, for each player i, strategies O_i have to

reach the minimum payoff of strategies in X_i^*. This suffices for an exact implementation if $O_i \subset X_i$, i.e. if there exists at least one strategy $x_i \notin O_i$. Otherwise, we determine whether it costs more to exceed the minimum constraint or the average constraint for all O_i if O_i covers whole columns and adjust V_i accordingly. Thus the algorithm ensures that only strategies in O are chosen while all strategies in O are selected.

The algorithm can be extended easily to work for instances where $X_i^* \subset O_i$. As the extension is straight-forward and does not provide any new insights, we omit it. The runtime of the algorithm can be determined to be $\mathrm{O}\big(n|X|^2\big)$, thus we can compute $k^*(O) = \max_{o \in O} V(o)$ and $k^*_{UNI}(O) = \varnothing_{o \in O} V(o)$ in polynomial time. □

Observe that these results imply that computing the (exact) implementation for both a worst-case and a uniform mechanism designer is in **P** for risk-averse players.

8.6 Round-Based Mechanisms

The previous sections dealt with static models only. Now, we extend our analysis to dynamic, round-based games, where the designer offers payments to the players after each round in order to make them change strategies. This opens many questions: For example, imagine a concrete game such as a *network creation game* [41] where all players are stuck in a costly Nash equilibrium. The goal of a mechanism designer could then be to guide the players into another, better Nash equilibrium. Many such extensions are reasonable; we present one model in more detail.

In a dynamic game, we regard a strategy profile as a state in which the participants find themselves. In a network context, each $x \in X$ could represent one particular network topology. We presume to find the game in an initial starting state $s^{T=0} \in X$ and that, in state $s^{T=t}$, each player i only sees the states she can reach by changing her strategy given the other players remain with their chosen strategies. Thus player i sees only strategy profiles in $X^{T=t}_{visible,i} = X_i \times \{s^{T=t}_{-i}\}$ in round t. In every round t, the mechanism designer offers the players a payment matrix $V^{T=t}$ (in addition to the game's static payoff matrix U). Then all players switch to their best visible strategy (which is any best response $B_i(s^{T=t}_{-i})$), and the game's state changes to $s^{T=t+1}$. Before the next round starts, the mechanism designer disburses the payments $V^{T=t}(s^{T=t+1})$ offered for the newly reached state. The same procedure is repeated until the mechanism designer decides to stop the game. See Figure 8.9 for an illustration.

We prove that a mechanism designer can guide the players to any strategy profile at zero cost in two rounds.

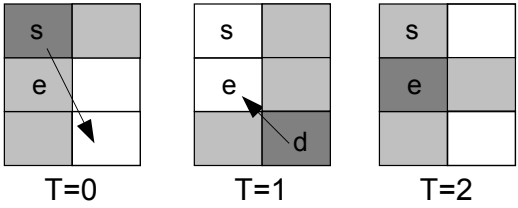


Figure 8.9: A dynamic game: Starting in **s**, strategy profile **e** can be implemented at zero cost within two rounds by taking a detour to **d**. The colored region marks the visible strategy profiles at each step.

Theorem 8.19. *Starting in an arbitrary strategy profile, a dynamic mechanism can be designed to lead the players to any strategy profile without any expenses in at most two rounds if* $|X_i| \geq 3 \ \forall i \in N$.

In order to simplify the proof we begin with a helper lemma. Let $X_{visible}^{T=t}$ denote the *visible strategy profile region* in round t, i.e.,

$$X_{visible}^{T=t} = \bigcup_{i=1}^{n} X_{visible,i}^{T=t}.$$

Lemma 8.20. *The third party can lead the players of a dynamic game to any strategy profile outside the visible strategy profile region without any expenses in one round.*

Proof. Let $s \in X$ be the starting strategy profile and e the desired end strategy profile in the non-visible region of s. The designer can implement e in just one round by offering each player i an infinite amount $V_i(x)$ for the strategy profile $x = (e_i, s_{-i})$ and zero for any other. Thus each player will switch to e_i. Since $V_i(e_i, s_{-i})$ are the only positive payments offered and since all $x = (e_i, s_{-i})$ are visible and e is non-visible from s, e is implemented at no cost. □

Proof of Theorem 8.19. Consider an arbitrary starting strategy profile s and a desired strategy profile e. If e is not visible from s, e is implementable at no cost in one round, as seen in Lemma 8.20. If e is visible from s, the interested party can still implement e for free by taking a detour to a strategy profile d which is neither in s' visible region nor in e's visible region. Such a strategy profile d exists if player i who sees e from s has at least 3 strategies to choose from and $|X_{-i}| \geq 2$. □

9

Leverage

This chapter studies to which extent the social welfare of a game can be influenced by an interested third party within economic reason, i.e., by taking the implementation cost into account. Besides considering classic, benevolent mechanism designers, we also analyze malicious mechanism designers. For instance, this chapter shows that a malicious mechanism designer can often corrupt games and worsen the players' situation to a larger extent than the amount of money invested. Surprisingly, no money is needed at all in some cases.

We provide algorithms for finding the leverage in games and show that for optimistic mechanism designers, computing the leverage or approximations thereof is NP-hard.

In this chapter, we analyze these problems' complexities and presents algorithms for finding the leverage of games for cautious and optimistic mechanism designers. We show that while the leverage of a single strategy profile can be computed efficiently for both cautious and optimistic mechanism designers, computing the uniform leverage of a game is **NP**-hard by a reduction from the SET COVER problem, and we provide a lower bound for the approximation attainable by any polynomial-time algorithm.

9.1 Worst-Case Leverage

Singletons

In the following we will examine a mechanism designer seeking to implement a game's best singleton, i.e., the strategy profile with the highest singleton leverage. Dually, a malicious designer attempts to find the profile of the largest malicious leverage.

We propose an algorithm that computes two arrays, LEV and $MLEV$, containing all (malicious) singletons' leverage within a strategy profile set O.

By setting $O = X$, the algorithm computes all singletons' (malicious) leverage of a game. We make use of an equation presented in [81] for computing a singleton's cost, namely that $k(z) = \sum_{i=1}^{n} \max_{x_i \in X_i} \{U_i(x_i, z_{-i}) - U_i(z_i, z_{-i})\}$.

Algorithm 9.1 Singleton (Malicious) Leverage

Input: Game G, Set $O \subseteq X$
Output: LEV and $MLEV$

```
 1: compute X*;
 2: for all strategy profiles x ∈ O do
 3:    lev[x] := − max_{x*∈X*} U(x*);
 4:    mlev[x] := min_{x*∈X*} U(x*);
 5: od
 6: for all Players i ∈ N do
 7:    for all x_{−i} ∈ O_{−i} do
 8:       m := max_{x_i∈X_i} U_i(x_i, x_{−i});
 9:       for all strategies z_i ∈ O_i do
10:          lev[z_i, x_{−i}] += 2 · U_i(z_i, x_{−i}) − m;
11:          mlev[z_i, x_{−i}] += U_i(z_i, x_{−i}) − 2m;
12:       od
13:    od
14: od
15: ∀ o ∈ O: LEV[o] := max{0, lev[o]};
16: ∀ o ∈ O: MLEV[o] := max{0, mlev[o]};
17: return LEV, MLEV;
```

Algorithm 9.1 initializes the *lev*-array with the negative value of the original game's maximal social gain in the non-dominated set and the *mlev*-array with
its minimal social gain. Next, it computes the set of non-dominated strategy profiles X^*; in order to do this, we check, for each player and for each of her strategies, whether the strategy is dominated by some other strategy. In the remainder, the algorithm adds up the players' contributions to the profiles' (malicious) leverage for each player and strategy profile. In any field z of the leverage array *lev*, we add the amount that Player i would contribute to the social gain if z was played and subtract the cost we had to pay her, namely $U_i(x_i, x_{-i}) - (m - U_i(x_i, x_{-i})) = 2U_i(x_i, x_{-i}) - m$. For any entry z in the malicious leverage array *mlev*, we subtract player i's contribution to the social gain and also twice the amount the designer would have to pay if z is played since she loses money and the players gain it, $-U_i(x_i, x_{-i}) - 2(m - U_i(x_i, x_{-i})) = U_i(x_i, x_{-i}) - 2m$. Finally, *lev* and *mlev* will contain all singletons' leverage and singletons' malicious leverage in O. By replacing the negative entries by zeros, the corresponding leverage arrays LEV and $MLEV$ are computed. The interested party can then lookup the best *non-negative* singleton by searching the maximal entry in the respective array.

Theorem 9.1. *For a game where every player has at least two strategies, Algorithm 9.1 computes all singletons' (malicious) leverage within a strategy profile set O in $\mathrm{O}\big(n|X|^2\big)$ time.*

Proof. The correctness of Algorithm 9.1 follows directly from the application of the (malicious) singleton leverage definition. It remains to prove the time complexity. Finding the non-dominated strategies in the original game requires time $\sum_{i=1}^{n} \binom{|X_i|}{2}|X_{-i}| = O(n|X|^2)$, and finding the maximal or minimal *gain* amongst the possible outcomes X^* of the original game requires time $\mathrm{O}(n\,|X|)$. The time for all other computations can be neglected asymptotically, and the claim follows. □

Strategy Profile Sets

Observe that implementing singletons may be optimal for entire strategy sets as well, namely in games where the strategy profile set yielding the largest (malicious) leverage is of cardinality 1. In some games, however, dominating all other strategy profiles in the set is expensive and unnecessary. Therefore, a mechanism designer is bound to consider sets consisting of more than one strategy profile as well to find a subset of X yielding the maximal (malicious) leverage. Moreover, we can construct games where the difference between the best (malicious) set leverage and the best (malicious) singleton leverage gets arbitrarily large. Figure 9.1 depicts such a game.

Although many factors influence a strategy profile set's (malicious) leverage, there are some simple observations. First, if rational players already choose strategies such that the strategy profile with the highest social gain is non-dominated, a designer will not be able to ameliorate the outcome. Just as well, a malicious interested party will have nothing to corrupt if a game already yields the lowest social gain possible.

Fact 9.2.

(i) *If a game G's social optimum $x_{opt} := \arg\max_{x \in X} U(x)$ is in X^* then $LEV(G) = 0$.*

(ii) *If a game G's social minimum $x_{worst} := \arg\min_{x \in X} U(x)$ is in X^* then $MLEV(G) = 0$.*

As an example, a class of games where both properties (i) and (ii) of Fact 9.2 always hold are *equal sum games*, where every strategy profile yields the same gain, $U(x) = c\ \forall x \in X, c : constant$. (Zero sum games are a special case of equal sum games where $c = 0$.)

Fact 9.3 (Equal Sum Games)**.** *The leverage and the malicious leverage of an equal sum game G is zero: $LEV(G) = 0$, $MLEV(G) = 0$.*

A well-known example of an zero sum game is *Matching Pennies*: Two players toss a penny. If both coins show the same face, player 2 gives his

$G =$

α / 0	1 / 0	γ / 0	γ / 0
1 / 0	α / 0	γ / 0	γ / 0
$\alpha - 1$ / 0	0 / $\alpha - 1$	0 / α	0 / 1
0 / $\alpha - 1$	$\alpha - 1$ / 0	0 / 1	0 / α

$V_O =$

0 / ∞	0 / ∞	0 / 0	0 / 0
0 / ∞	0 / ∞	0 / 0	0 / 0
1 / 1	1 / 1	∞ / 0	∞ / 0
1 / 1	1 / 1	∞ / 0	∞ / 0

$V_S =$

0 / ∞	0 / 0	0 / 0	0 / 0
0 / ∞	0 / 0	0 / 0	0 / 0
0 / α	0 / 0	0 / 0	0 / 0
α / 1	0 / α	∞ / 0	∞ / 0

Figure 9.1: Two-player game where the set O bears the largest leverage. Implementation V_O yields $X^*(V_O) = O$ and V_S yields one non-dominated strategy profile. By offering payments V_O, a mechanism designer has cost 2, no matter which $o \in O$ will be played. However, she changes the social welfare to $\alpha - 1$. If $\gamma < \alpha - 3$ then O has a leverage of $\alpha - 3 - \gamma$ and if $\gamma > \alpha + 3$ then O has a malicious leverage of $\gamma - \alpha - 3$. Any singleton $o \in O$ has an implementation cost of $\alpha + 1$, yet the resulting leverage is 0 and the malicious leverage is $\max(0, \gamma - 3\alpha - 1)$. This demonstrates that a profile set O's (malicious) leverage can be arbitrarily large, even if its singletons's (malicious) leverage is zero.

penny to player 1; if the pennies do not match, player 2 gets the pennies. This matching pennies game features another interesting property: There is no dominated strategy. Therefore an interested party could only implement strategy profile sets O which are subsets of X^*. This raises the question whether a set $O \subseteq X^*$ can ever have a (malicious) leverage. We find that the answer is no and moreover:

Theorem 9.4. *The leverage of a strategy profile set $O \subseteq X$ intersecting with the set of non-dominated strategy profiles X^* is $(M)LEV = 0$.*

Proof. Assume that $|O \cap X^*| > 0$ and let $\hat{z}$ be a strategy profile in the intersection of O and X^*. Let $x^*_{max} := \arg\max_{x^* \in X^*} U(x^*)$ and $x^*_{min} := \arg\min_{x^* \in X^*} U(x^*)$. Let V_{LEV} be any implementation of O reaching $LEV(O)$ and V_{MLEV} any implementation of O reaching $MLEV(O)$. We get for the leverage

$$\begin{aligned} LEV(O) &= \max\{0, \min_{z \in X^*(V_{LEV})} \{U(z) - V_{LEV}(z)\} - U(x^*_{max})\} \\ &\leq \max\{0, [U(\hat{z}) - V_{LEV}(\hat{z})] - U(x^*_{max})\} \\ &\leq \max\{0, U(x^*_{max}) - V_{LEV}(\hat{z}) - U(x^*_{max})\} \\ &= \max\{0, -V_{LEV}(\hat{z})\} \\ &= 0, \end{aligned}$$

and for the malicious leverage

$$\begin{aligned} MLEV(O) &= \max\{0, U(x^*_{min}) - \max_{z \in X^*(V_{MLEV})} [U(z) + 2V_{MLEV}(z)]\} \\ &\leq \max\{0, U(x^*_{min}) - U(\hat{z}) - 2V_{MLEV}(\hat{z})\} \\ &\leq \max\{0, U(x^*_{min}) - U(x^*_{min}) - 2V_{MLEV}(\hat{z})\} \\ &= \max\{0, -2V_{MLEV}(\hat{z})\} \\ &= 0. \end{aligned}$$

□

In general, the problem of computing a strategy profile set's (malicious) leverage seems computationally hard. It is related to the problem of computing a set's implementation cost, which is conjectured to be **NP**-hard in the previous chapter, and hence, we conjecture the problem of finding $LEV(O)$ or $MLEV(O)$ to be **NP**-hard in general as well. In fact, we can show that computing the (malicious) leverage has at least the same complexity as computing a set's cost.

Theorem 9.5. *If the computation of a set's implementation cost is* **NP***-hard, then the computation of a strategy profile set's (malicious) leverage is also* **NP***-hard.*

Proof. We proceed by reducing the problem of computing $k(O)$ to the problem of computing $MLEV(O)$. Theorem 9.4 allows us to assume that O and X^* do not intersect since $O \cap X^* \neq \emptyset$ implies $MLEV(O) = 0$. By definition, a strategy profile set's cost are $k(O) = \min_{V \in \mathcal{V}(O)} \{\max_{z \in X^*(V)} V(z)\}$ and from the malicious leverage's definition, we have $\min_{V \in (V)} \{\max_{z \in X^*(V)} \{U(z) + 2V(z)\}\} = \min_{x^* \in X^*} U(x^*) - mlev(O)$. The latter equation's left hand side almost matches the definition of $k(O)$ if not for the term $U(z)$ and a factor

of 2. If we can modify the given game such that all strategy profiles inside $X^*(V) \subseteq O$ have a gain of

$$\gamma := -2n \max_{x \in X}\{\max_{i \in N} U_i(x)\} - \min_{x^* \in X^*} U(x^*) - \epsilon$$

for some $\epsilon > 0$, we will be able to reduce O's cost to

$$\begin{aligned} k(O) &= \frac{\min_{x^* \in X^*} U(x^*) - mlev(O) - \gamma}{2} \\ &= -mlev(O) + 2n \max_{x \in X}\{\max_{i \in N} U_i(x)\} + \epsilon, \end{aligned}$$

thus $mlev(O) > 0$ and $MLEV(O) = mlev(O)$, ensuring that $MLEV(O)$ and $mlev(O)$ are polynomially reducible to each other. This is achieved by the following transformation of a problem instance (G, O) into a problem instance (G', O): Add an additional Player $n + 1$ with one strategy a and a payoff function $U_{n+1}(x)$ equal to $\gamma - U(x)$ if $x \in O$ and 0 otherwise. Thus, a strategy profile x in G' has social gain equal to γ if it is in O and equal to $U(x)$ in the original game if it is outside O. As Player $n + 1$ has only one strategy available, G' has the same number of strategy profiles as G and furthermore, there will be no payments V_{n+1} needed in order to implement O. Player $(n + 1)$'s payoffs impact only the profiles' gain, and they have no effect on how the other players decide their tactics. Thus, the non-dominated set in G' is the same as in G and it does not intersect with O. Since the transformation does not affect the term $\min_{x^* \in X^*} U(x^*)$, the set's cost in G are equal to $(\min_{x^* \in X^*} U(x^*) - MLEV(O) - \gamma)/2$ in G'.

Reducing the problem of computing $k(O)$ to $lev(O)$ is achieved by using the same game transformation where an additional player is introduced such that $\forall o \in O : U(o) = \gamma$, where

$$\gamma := n \max_{x \in X}\{\max_{i \in N}\{U_i(x)\}\} + \max_{x^* \in X^*}\{U(x^*)\} + \epsilon$$

for some $\epsilon > 0$. We can then simplify the equation to

$$lev(O) = \gamma - k(O) - \max_{x^* \in X^*} U(x^*) = n \max_{x \in X}\{\max_{i \in N}\{U_i(x)\}\} - k(O) + \epsilon > 0$$

and thus we find the cost $k(O)$ by computing

$$n \max_{x \in X}\{\max_{i \in N}\{U_i(x)\}\} - LEV(O) - \epsilon.$$

□

The task of finding a strategy profile set's leverage is computationally hard. Recall that we have to find an implementation V of O which maximizes the term $\min_{z \in X^*(V)}\{U(z) - V(z)\}$. Thus, there is at least one implementation $V \in \mathcal{V}(O)$ bearing O's leverage. Since this V implements a subset

Algorithm 9.2 Exact Leverage

Input: Game G, convex set O with $O_{-i} \subset X_{-i} \forall\ i$
Output: $LEV^*(O)$

$V_i(x) := 0$, $W_i(x) := 0\ \forall x \in X$, $i \in N$;
$V_i(o_i, \bar{o}_{-i}) := \infty\ \forall i \in N$, $o_i \in O_i$, $\bar{o}_{-i} \in X_{-i} \backslash O_{-i}$;
compute X_i^*;
return $\max\{0, ExactLev(V, n) - \max_{x^* \in X^*} U(x^*)\}$;

***ExactLev(V , i)*:**
Input: payments V, current player i
Output: $lev^*(O)$ for $G(V)$

if $\left|X_i^*(V) \backslash O_i\right| > 0$ **then**
  $s :=$ any strategy in $X_i^*(V) \backslash O_i$; $lev_{best} := 0$;
  for all $o_i \in O_i$ **do**
    for all $o_{-i} \in O_{-i}$ **do**
      $W_i(o_i, o_{-i}) := \max\{0, U_i(s, o_{-i}) - (U_i(o_i, o_{-i}) + V_i(o_i, o_{-i}))\}$;
    od
    $lev := ExactLev(V + W, i)$;
    if $lev > lev_{best}$ **then**
      $lev_{best} := lev$;
    fi
    for all $o_{-i} \in O_{-i}$ **do**
      $W_i(o_i, o_{-i}) := 0$;
    od
  od
  return lev_{best};
fi
if $i > 1$ **return** $ExactLev(V, i-1)$;
else return $\min_{o \in O}\{U(o) - V(o)\}$;

of O exactly, it is also valid to compute O's leverage by searching among all subsets O' of O the one with the largest exact leverage $LEV^*(O')$.[1]

In the following we will provide an algorithm which computes a convex strategy profile set's exact leverage. It makes use of the fact that if $X^*(V)$ has to be a subset of O, each strategy $\bar{o}_i \notin O_i$ must be dominated by at least one strategy o_i in the resulting game $G(V)$ – a property which has been observed and exploited before in the previous chapter in order to compute a set's exact cost. In order to compute $LEV(O)$, we can apply Algorithm 9.2 for all convex subsets and return the largest value found.

Theorem 9.6. *Algorithm 9.2 computes a strategy profile set's exact leverage in time*

$$\mathrm{O}\left(|X|^2 \max_{i \in N}(|O_i|^{n|X_i^* \backslash O_i|-1}) + n|O| \max_{i \in N}(|O_i|^{n|X_i^* \backslash O_i|})\right).$$

[1] Note that we do not provide algorithms for computing the malicious leverage but for the benevolent leverage only. However, we are sure that a malicious mechanism designer will figure out how to adapt our algorithms for the benevolent leverage for a nastier purpose.

Proof. Clearly, the algorithm is correct as it searches for all possibilities of a strategy in $X_i \backslash O_i$ to be dominated by a strategy in O_i. The time complexity follows from solving the doubly recursive equation over the strategy set and the number of players (compare to the analysis of Algorithm 8.2 in the previous chapter). □

9.2 Uniform Leverage

A mechanism designer calculating her average case cost is more optimistic than an anxious designer. Thus, the observation stating that the uniform (malicious) leverage is always at least as large as the worst-case (malicious) leverage does not surprise.

Theorem 9.7. *A set's uniform (malicious) leverage is always larger than or equal to the set's (malicious) leverage.*

Proof.

$$\begin{aligned} lev_{UNI}(O) &= \max_{V \in \mathcal{V}(O)} \{ \underset{z \in X^*(V)}{\varnothing} \{U(z) - V(z)\} \} - \underset{x^* \in X^*(V)}{\varnothing} U(x^*) \\ &\geq \max_{V \in \mathcal{V}(O)} \{ \min_{z \in X^*(V)} \{U(z) - V(z)\} \} - \max_{x^* \in X^*(V)} U(x^*) \\ &= lev(O) \end{aligned}$$

and

$$\begin{aligned} mlev_{UNI}(O) &= \underset{x^* \in X^*(V)}{\varnothing} U(x^*) - \min_{V \in \mathcal{V}(O)} \{ \underset{z \in X^*(V)}{\varnothing} \{U(z) + 2V(z)\} \} \\ &\geq \min_{x^* \in X^*(V)} \{U(x^*)\} - \min_{V \in \mathcal{V}(O)} \{ \max_{z \in X^*(V)} \{U(z) + 2V(z)\} \} \\ &= mlev(O). \end{aligned}$$

□

Another difference concerns the sets O intersecting with X^*, i.e., cases where $O \cap X^* \neq \emptyset$: Unlike the worst-case leverage (Theorem 9.4), the uniform leverage can exceed zero in these cases, as can be verified by calculating O's leverage in Figure 3.

Complexity

We show how the uniform implementation cost can be computed in polynomial time given the corresponding leverage. Thus the complexity of computing the leverage follows from the **NP**-hardness of finding the optimal implementation cost. The lower bounds are derived by modifying the games constructed from the SET COVER problem in Theorem 8.11, and by using a lower bound for the approximation quality of the SET COVER problem.

If no polynomial time algorithm can approximate the size of a set cover within a certain factor, we get an arbitrarily small approximated leverage $LEV_{UNI}^{approx} \leq \epsilon$ while the actual leverage is large. Hence the approximation ratio converges to infinity and, unless **P**=**NP**, there exists no polynomial time algorithm approximating the leverage of a game within any function of the input length.

Theorem 9.8. *For games with at least two players, the problem of*

- *computing a strategy profile set's* exact uniform leverage *as well as*
- *computing a its* exact malicious uniform leverage *are* **NP***-hard.*

Furthermore, the (exact) uniform leverage of O cannot be approximated in polynomial time within any function of the input length unless **P**=**NP**.

Proof.
NP*-Hardness: Exact Case.* The claim follows from the observation that if $(M)LEV_{UNI}^{*}(O)$ is found, we can immediately compute $k_{UNI}^{*}(O)$ which is **NP**-hard (Theorem 8.11). Due to the fact that any $z \in O$ is also in $X^*(V)$ for any $V \in \mathcal{V}^*(O)$ we know that

$$\begin{aligned} lev_{UNI}(O) &= \max_{V \in \mathcal{V}^*(O)} \{ \underset{z \in X^*(V)}{\varnothing} \{U(z) - V(z)\} \} - \underset{z \in X^*}{\varnothing} U(x^*) \\ &= \max_{V \in \mathcal{V}^*(O)} \{ \underset{z \in X^*(V)}{\varnothing} U(z) - \underset{z \in X^*(V)}{\varnothing} V(z) \} - \underset{x^* \in X^*}{\varnothing} U(x^*) \\ &= \underset{z \in X^*(V)}{\varnothing} U(z) - \min_{V \in \mathcal{V}^*(O)} \{ \underset{z \in X^*(V)}{\varnothing} V(z) \} - \underset{x^* \in X^*}{\varnothing} U(x^*) \\ &= \underset{z \in X^*(V)}{\varnothing} U(z) - \mathbf{k}_{\mathbf{UNI}}^{*}(\mathbf{O}) - \underset{x^* \in X^*}{\varnothing} U(x^*) \end{aligned}$$

and

$$\begin{aligned} mlev_{UNI}(O) &= \underset{x^* \in X^*}{\varnothing} U(x^*) - \min_{V \in \mathcal{V}^*(O)} \{ \underset{z \in X^*(V)}{\varnothing} \{U(z) + 2V(z)\} \} \\ &= \underset{x^* \in X^*}{\varnothing} U(x^*) - \underset{z \in X^*(V)}{\varnothing} U(z) - 2 \min_{V \in \mathcal{V}^*(O)} \{ \underset{z \in X^*(V)}{\varnothing} V(z) \} \\ &= \underset{x^* \in X^*}{\varnothing} U(x^*) - \underset{z \in X^*(V)}{\varnothing} U(z) - 2\mathbf{k}_{\mathbf{UNI}}^{*}(\mathbf{O}). \end{aligned}$$

Observe that $\varnothing_{x^* \in X^*} U(x^*)$ and $\varnothing_{z \in X^*(V)} U(z)$ can be computed easily. Moreover, as illustrated in the proof of Theorem 9.5, we can efficiently construct a problem instance (G', O) from any (G, O) with the same cost, such that for G': $(m)lev_{(UNI)} = (M)LEV_{(UNI)}$.

Non-Exact Case. The claim can be proved by reducing the **NP**-hard problem of computing $k_{UNI}(O)$ to the problem of computing $(M)LEV_{UNI}(O)$. This reduction uses a slight modification of player 3's utility in the respective game in the proof of Theorem 8.11 ensuring $\forall z \in O\ U(z) = \gamma$, where

$$\gamma := -4(m+l)^2 - 2m^2 + m(l+m).$$

Set

$$\begin{aligned}
&U_3(s_i, e_j, a) := \gamma - U_1(s_i, e_j, a) - U_2(s_i, e_j, a),\\
&U_3(s_i, e_j, b) := \gamma + 2 - U_1(s_i, e_j, a) - U_2(s_i, e_j, a),\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{for } i \in [1, m],\ j \in [1, l]\\
&U_3(s_i, s_j, a) := \gamma - U_1(s_i, s_j, a) - U_2(s_i, s_j, a),\\
&U_3(s_i, s_j, b) := \gamma + 2 - U_1(s_i, s_j, a) - U_2(s_i, s_j, a),\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{for } i \neq j\\
&U_3(s_i, s_i, a) := \gamma - U_1(s_i, s_i, a) - U_2(s_i, s_i, a),\\
&U_3(s_i, s_i, b) := \gamma + (m + l + 2) - U_1(s_i, s_i, a) - U_2(s_i, s_i, a),\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{for all } i
\end{aligned}$$

Since in this 3-player game, $mlev_{UNI}(O) > 0$, $k_{UNI}(O)$ depends only on O's (malicious) leverage and the average social gain, namely

$$k_{UNI}(O) = \underset{x^* \in X^*}{\varnothing} (U(x^*) - MLEV_{UNI}(O))/2.$$

Thus, once $MLEV_{UNI}(O)$ is known, $k_{UNI}(O)$ can be computed immediately, and therefore finding the uniform malicious leverage is **NP**-hard as well. We can adapt this procedure for $LEV_{UNI}(O)$ as well.

Lower Bound: Exact Case. The game constructed from the SET COVER problem in Theorem 8.11 for the exact case is modified as follows: The utilities of player 1 remain the same. The utilities of player 2 are all zero except for $U_2(e_1, r) := (l+m)(\sum_{i=1}^{m} |S_i|(m+1)/(ml+m) - k\mathcal{LB} - \epsilon)$, where k is the minimal number of sets needed to solve the corresponding SET COVER instance, $\epsilon > 0$, and $\mathcal{LB}$ denotes a lower bound for the approximation quality of the SET COVER problem. Observe that X^* consists of all strategy profiles of column r. The target set we want to implement exactly is given by $O_1 = \{s_1, ..., s_m\}$ and $O_2 = \{e_1, ..., e_l, d\}$. We compute

$$\begin{aligned}
lev_{UNI}^{opt} &= \underset{o \in O}{\varnothing} U(o) - \underset{x \in X^*}{\varnothing} U(x) - k\\
&= \sum_{i=1}^{m} |S_i|(m+1)/(ml+m) - \sum_{i=1}^{m} |S_i|(m+1)/(ml+m)\\
&\quad -(-k\mathcal{LB} - \epsilon) - k\\
&= k(\mathcal{LB} - 1) + \epsilon.
\end{aligned}$$

As no polynomial time algorithm can approximate k within a factor $\mathcal{LB}$, it holds that $LEV_{UNI}^{approx} \leq \epsilon$. Thus the claim follows for a benevolent mechanism designer, because $\lim_{\epsilon \to 0} LEV_{UNI}^{opt}/LEV_{UNI}^{approx} = \infty$.

For malicious mechanism designers, we modify the utilities of the game from the proof of Theorem 8.12 for player 2:

$$U_2(e_1, r) := (l+m)\left(2k\mathcal{LB} + \epsilon + \frac{\sum_{i=1}^{m}|S_i|(m+1)}{ml+m}\right),$$

and

$$U_2(\_, \_) := 0 \qquad \text{for all other profiles.}$$

It is easy to see that by a similar analysis as performed above, the theorem also follows in this case.

Lower Bound: Non-Exact Case. We modify the game construction of Theorem 8.11's proof for the non-exact case. Let

$$\gamma := \frac{\sum_{i=1}^{m}|S_i|(m+l)^2 + m^2(m+l)^2 + 3m(m+l)}{m^2+ml+m}.$$

For benevolent designers, we set

$$U_2(e_1, r, b) := (m+l)\left(\gamma - k\mathcal{LB} - \epsilon\right),$$

where k is the minimal number of sets needed to solve the corresponding SET COVER instance, $\epsilon > 0$, and $\mathcal{LB}$ denotes a lower bound for the approximation quality of the SET COVER problem and zero otherwise. Observe that O has not changed, $X^* = \{x | x \in X, x = (\_, r, b)\}$, and payments outside O do not contribute to the implementation cost; therefore, implementing O exactly is still the cheapest solution. By a similar analysis as in the proof of Theorem 8.11 the desired result is attained. For malicious mechanism designers we can proceed as above if we set

$$U_2(e_1, r, b) := (m+l)\left(\gamma + 2k\mathcal{LB} + \epsilon\right).$$

□

Algorithms

To round off our analysis of the uniform (malicious) leverage, we investigate algorithms for risk neutral mechanism designers. Recall Algorithm 9.1 in Section 9.1 which computes the leverage of singletons of a desired strategy profile set. We can adapt this algorithm in Line 3 and 4 to conform to the definition of the uniform leverage, i.e., set $mlev[x] := \varnothing_{x^* \in X^*} U(x^*)$ and $mlev[x] := -mlev[x]$. This algorithm thus helps to find an optimal singleton.

A benevolent mechanism designer can adapt Algorithm 9.2 in order to compute $LEV^*_{UNI}(O)$: She only has to change Line 4 to

$$\textbf{return} \max\{0, ExactLev(V, n) - \underset{x^* \in X^*}{\varnothing} U(x^*)\}$$

and 'min' in Line 13 of *ExactLev* to '$\varnothing$'. Invoking this algorithm for any $O' \subseteq O$ yields the subset O with maximal leverage $LEV_{UNI}(O)$.

Due to Theorem 9.8 there is no polynomial time algorithm giving a non-trivial approximation of a uniform leverage. The simplest method to find a lower bound for $LEV_{UNI}(O)$ is to search the singleton in O with the largest uniform leverage. Unfortunately, there are games (cf. Figure 8.4) where this lower bound is arbitrarily bad, as it is for the worst case leverage.

10

Conclusions and Outlook

There is an increasing demand for a game theoretic analysis and incentive-compatible solutions for distributed computing systems such as the Internet. This is mainly due to the fact that they consist of different stake-holders aiming at maximizing their individual profits. Rendering distributed systems resilient to non-cooperative behavior has become an important research topic. This part of the thesis has raised the following question: Which outcomes can be implemented by promising players money while the eventual payments are bounded? We have presented algorithms for various objectives yielding implementations of low cost as well as computational complexity results for cautious and optimistic mechanism designers. Moreover, we have initiated the study of risk-averse players and round-based games and shown that efficient algorithms do exist for these scenarios. In addition, we have studied benevolent and malicious mechanism designers intending to change the game's outcome if the improvement or deterioration in social welfare exceeds the implementation cost. Our results are summarized in Figure 10 and Figure 10.

Our models still offer interesting questions to be tackled in future research, including the quest for implementation cost approximation algorithms or for game classes which allow a leverage approximation. Furthermore, the mixed leverage and the leverage of dynamic games with an interested third party offering payments in each round are still unexplored.

Of course, we have to be aware of the limitations of these models and results as well. There are hardly any situations where all the involved parties can be determined, let alone the exact utility of each participant for each outcome. Without this knowledge however, neither our benevolent nor our malicious mechanism designer is able to compute an optimal payment distribution of its available incentives.

Impl. Cost	Complexity	Properties
Uniform	NP-complete SINGLETON $\mathrm{O}\big(n \cdot \sum_i \lvert X_i \rvert\big)$ ZERO $\mathrm{O}\big(n \lvert X \rvert^2\big)$	NE 0-implementable
Worst-case	conjectured to be NP-complete SINGLETON $\mathrm{O}\big(n \cdot \sum_i \lvert X_i \rvert\big)$ ZERO $\mathrm{O}\big(n \lvert X \rvert^2\big)$	NE 0-implementable
Risk-averse	$\mathrm{O}\big(n \lvert X \rvert^2\big)$	Singletons cheapest
Average payoff	All regions 0-implementable	
Round-based	Singletons 0-implementable in 2 rounds	

Figure 10.1: Complexity results for the computation of the implementation cost. Unless stated otherwise, complexities refer to the problem of computing any strategy profile's implementation cost. SINGLETON indicates the complexity of computing a singleton's implementation cost. ZERO indicates the complexity of deciding for a strategy profile region whether it is 0-implementable.

Leverage	Complexity	Properties
Uniform	NP-complete SINGLETON $\mathrm{O}\big(n \cdot \sum_i \lvert X_i \rvert\big)$	$MLEV_{UNI} \geq MLEV$
Worst-case	as hard as implementation cost SINGLETON $\mathrm{O}\big(n \cdot \sum_i \lvert X_i \rvert\big)$	$O \cap X^* \neq \emptyset$ $\Rightarrow (M)LEV = 0$ social opt/worst $\in X^*$ $\Rightarrow (M)LEV = 0$ Equal-sum games $\Rightarrow (M)LEV = 0$

Figure 10.2: Complexity results for the computation of the leverage. SINGLETON indicates the complexity of computing a singleton's leverage.

Moreover, a situation might change unexpectedly resulting in adapted utilities of the players and the carefully calculated payments will not lead to the desired outcome. Game theory is based on behavioral assumptions on the nature of the players. However, one does never know for sure how powerful the players are and how much they are prepared to invest in finding out their optimal strategies. These drawbacks are common to any model, since a model can by definition only try to capture some of the fundamental characteristics and aspects of a system, hoping to allow for a rigorous analysis and best possible solutions. As a consequence, we believe that our insights on mechanism design by creditability can provide some useful intuition and guidance in designing distributed systems.

Bibliography

[1] J. Aein. Power Balancing in Systems Employing Frequency Reuse. *COMSAT Technical Revue*, 3(2):277–300, 1973.

[2] A. S. Aiyer, L. Alvisi, A. Clement, M. Dahlin, J.-P. Martin, and C. Porth. BAR Fault Tolerance for Cooperative Services. In *Proceedings of the 20th ACM Symposium on Operating Systems Principles (SOSP)*, pages 45–58, 2005.

[3] S. Albers and H. Bals. Dynamic TCP Acknowledgement: Penalizing Long Delays. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 47–55, 2003.

[4] G. Alnifie and R. Simon. A Multi-channel Defense Against Jamming Attacks in Wireless Sensor Networks. In *Proceedings of the 3rd ACM Workshop on QoS and Security for Wireless and Mobile Networks (Q2SWINET)*, pages 95–104, 2007.

[5] N. Alon, D. Moshkovitz, and M. Safra. Algorithmic Construction of Sets for k-Restrictions. *ACM Transactions on Algorithms (TALG)*, pages 153–177, 2006.

[6] G. Alonso, E. Kranakis, C. Sawchuk, R. Wattenhofer, and P. Widmayer. Randomized Protocols for Node Discovery in Ad-hoc Multichannel Broadcast Networks. In *Proceedings of the 2nd Conference on Adhoc Networks and Wireless (ADHOCNOW)*, volume 2865, pages 104–115, 2003.

[7] G. Alonso, E. Kranakis, R. Wattenhofer, and P. Widmayer. Probabilistic Protocols for Node Discovery in Ad-Hoc, Single Broadcast Channel Networks. In *Proceedings of the 17th International Symposium on Parallel and Distributed Processing (IPDPS)*, page 8, 2003.

[8] K. Arrow. *Social Choice and Individual Values*. Wiley, 1951.

[9] J. Aspnes. Competitive analysis of distributed algorithms. *Online Algorithms: The State of the Art, Lecture Notes in Computer Science; Vol. 1442*, pages 118–146, 1998.

[10] J. Aspnes, K. Chang, and A. Yampolskiy. Inoculation Strategies for Victims of Viruses and the Sum-Of-Squares Partition Problem. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 43–52, 2005.

[11] B. Awerbuch, A. Richa, and C. Scheideler. A Jamming-Resistant MAC Protocol for Single-Hop Wireless Networks. In *Proceedings of the 27th Symposium on Principles of Distributed Computing (PODC)*, pages 45–54, 2008.

[12] M. Babaioff, M. Feldman, and N. Nisan. Combinatorial Agency. In *Proceedings of the 7th ACM Conference on Electronic Commerce (EC)*, pages 18–28, 2006.

[13] M. Babaioff, R. Kleinberg, and C. H. Papadimitriou. Congestion Games with Malicious Players. In *Proceedings of the 8th ACM Conference on Electronic Commerce (EC)*, pages 103–112, 2007.

[14] H. Baldus, K. Klabunde, and G. Müsch. Reliable set-up of medical body-sensor networks. *Lecture Notes in Computer Science*, pages 353–363, 2004.

[15] E. Bayraktaroglu, C. King, X. Liu, G. Noubir, R. Rajaraman, and B. Thapa. On the Performance of IEEE 802.11 under Jamming. In *Proceedings of the 27th Annual Joint Conference. of the IEEE Computer and Communications Societies (INFOCOM)*, pages 1265–1273, 2008.

[16] L. Becchetti, P. Korteweg, A. Marchetti-Spaccamela, M. Skutella, L. Stougie, and A. Vitaletti. Latency Constrained Aggregation in Sensor Networks. In *Proceedings of the Annual European Symposium on Algorithms (ESA)*, pages 88–99, 2006.

[17] A. Behzad and I. Rubin. On the Performance of Graph-based Scheduling Algorithms for Packet Radio Networks. In *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM)*, volume 6, 2003.

[18] A. Behzad and I. Rubin. Impact of Power Control on the Performance of Ad Hoc Wireless Networks. In *Proceedings of the 24^{th} Annual Joint Conference of the IEEE Computer and Communication Societies (INFOCOM)*, 2005.

[19] D. Bertsekas and R. Gallager. *Data Networks*. Prentice Hall, 1992.

[20] P. Björklund, P. Värbrand, and D. Yuan. A Column Generation Method for Spatial TDMA Scheduling in Ad Hoc Networks. *Ad Hoc Networks*, 2(4):405–418, 2004.

[21] S. Borbash and A. Ephremides. Wireless Link Scheduling With Power Control and SINR Constraints. *IEEE Transactions on Information Theory*, 52(5):5106–5111, 2006.

[22] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[23] G. Brar, D. Blough, and P. Santi. Computationally Efficient Scheduling with the Physical Interference Model for Throughput Improvement in Wireless Mesh Networks. In *Proceedings of the 12th International Conference on Mobile Computing and Networking (MOBICOM)*, pages 2–13, 2006.

[24] C. Brito, E. Koutsoupias, and S. Vaya. Competitive Analysis of Organization Networks or Multicast Acknowledgement: How Much to Wait? In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 627–635, 2004.

[25] N. Burri, P. von Rickenbach, and R. Wattenhofer. Dozer: Ultra-Low Power Data Gathering in Sensor Networks. In *Proceedings of the 6th International Conference on Information Processing in Sensor Networks (IPSN)*, pages 450–459, 2007.

[26] J. T. Chiang and Y.-C. Hu. Cross-layer Jamming Detection and Mitigation in Wireless Broadcast Networks. In *Proceedings of the 13th ACM Conference on Mobile Computing and Networking (MOBICOM)*, pages 346–349, 2007.

[27] G. Christodoulou and E. Koutsoupias. The Price of Anarchy of Finite Congestion Games. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC)*, pages 67–73, 2005.

[28] R. Cole, Y. Dodis, and T. Roughgarden. How Much Can Taxes Help Selfish Routing? In *Proceedings of the 4th ACM conference on Electronic commerce (EC)*, pages 98–107, 2003.

[29] R. Cole, Y. Dodis, and T. Roughgarden. Pricing Network Edges for Heterogeneous Selfish Users. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC)*, pages 521–530, 2003.

[30] C. W. Commander, P. M. Pardalos, V. Ryabchenko, S. Uryasev, and G. Zrazhevsky. The Wireless Network Jamming Problem. In *Air Force Research Laboratory, Tech Report 07-11-06-332*, 2007.

[31] D. Cook and S. Das. How Smart are our Environments? An Updated Look at the State of the Art. *Pervasive and Mobile Computing*, 3(2):53–73, 2007.

[32] R. L. Cruz and A. V. Santhanam. Optimal Routing, Link Scheduling and Power Control in Multi-hop Wireless Networks. In *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2003.

[33] R. K. Dash, N. R. Jennings, and D. C. Parkes. Computational-Mechanism Design: A Call to Arms. *IEEE Intelligent Systems*, pages 40–47, 2003. Special Issue on Agents and Markets.

[34] S. Dolev, S. Gilbert, R. Guerraoui, and C. Newport. Gossiping in a Multi-Channel Radio Network (An Oblivious Approach to Coping With Malicious Interference). In *Proceedings of the 21st Annual Conference on Distributed Computing (DISC)*, volume 4731, page 208, 2007.

[35] S. Dolev, S. Gilbert, R. Guerraoui, and C. Newport. Secure Communication over Radio Channels. In *Proceedings of the 27th Symposium on Principles of Distributed Computing (PODC)*, pages 105–114, 2008.

[36] D. R. Dooly, S. A. Goldman, and S. D. Scott. TCP Dynamic Acknowledgment Delay: Theory and Practice. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC)*, pages 389–398, 1998.

[37] H. Dubois-Ferrier, R. Meier, L. Fabre, and P. Metrailler. TinyNode: A Comprehensive Platform for Wireless Sensor Network Applications. In *Proceedings of the 5th international conference on Information processing in sensor networks*, pages 358–365, 2006.

[38] T. A. ElBatt and A. Ephremides. Joint Scheduling and Power Control for Wireless Ad-hoc Networks. In *Proceedings of the Annual Joint Conference of the IEEE Computer and Communication Societies (INFOCOM)*, 2002.

[39] A. Ephremides and T. V. Truong. Scheduling Broadcasts in Multihop Radio Network. *IEEE Transactions on Communications*, 38(4):456–460, 1990.

[40] L. Evers, M. Bijl, M. Marin-Perianu, R. Marin-Perianu, and P. Havinga. *Wireless Sensor Networks and Beyond: A Case Study on Transport and Logistics*. Centre for Telematics and Information Technology, University of Twente, 2005.

[41] A. Fabrikant, A. Luthra, E. Maneva, C. H. Papadimitriou, and S. Shenker. On a Network Creation Game. In *Proceedings of the 22nd Annual Symposium on Principles of Distributed Computing(PODC)*, pages 347–351, 2003.

[42] U. Feige. A Threshold of $\log n$ for Approximating Set Cover. *Journal of the ACM*, pages 634–652, 1998.

[43] J. Feigenbaum, C. Papadimitriou, R. Sami, and S. Shenker. A BGP-based Mechanism for Lowest-Cost Routing. In *Proceedings of the 21st Annual Symposium on Principles of Distributed Computing (PODC)*, pages 173–182, 2002.

[44] G. J. Foschini and Z. Miljanic. A Simple Distributed Autonomous Power Control Algorithms and its Convergence. *IEEE Transactions on Vehicular Technology*, 40(4):641–646, 1993.

[45] J. S. Frederiksen and K. S. Larsen. Packet Bundling. In *Proceedings of the 8th Scandinavian Workshop on Algorithm Theory (SWAT)*, pages 328–337, 2002.

[46] M. Fussen, R. Wattenhofer, and A. Zollinger. Interference Arises at the Receiver. In *International Conference on Wireless Networks, Communications, and Mobile Computing (WIRELESSCOM)*, volume 1, 2005.

[47] S. Gilbert, R. Guerraoui, and C. Newport. Of Malicious Motes and Suspicious Sensors. In *Proceedings of the Conference on Principles of Distributed Systems (OPODIS)*, volume 4305, page 215, 2006.

[48] O. Goussevskaia, T. Moscibroda, and R. Wattenhofer. Local Broadcasting in the Physical Interference Model. In *Proceedings of the 5th International Workshop on Foundations of Mobile Computing (DIALM-POMC)*, pages 35–44, 2008.

[49] S. Grandhi, R. Vijayan, and D. Goodman. Distributed Power Control in Cellular Radio Systems. *Communications, IEEE Transactions on*, 42(234 Part 1):226–228, 1994.

[50] J. Grönkvist. *Interference-Based Scheduling in Spatial Reuse TDMA*. PhD thesis, Royal Institute of Technology, Stockholm, Sweden, 2005.

[51] J. Grönkvist and A. Hansson. Comparison Between Graph-Based and Interference-Based STDMA Scheduling. In *Proceedings of the 2nd ACM International Symposium on Mobile Ad Hoc Networking & Computing (MOBIHOC)*, pages 255–258, 2001.

[52] P. Gupta and P. R. Kumar. Critical Power for Asymptotic Connectivity in Wireless Networks. *Stochastic Analysis, Control, Optimization and Applications: A Volume in Honor of W. H. Fleming (March 1998)*, pages 547–566, 1998.

[53] P. Gupta and P. R. Kumar. The Capacity of Wireless Networks. *IEEE Transactions of Information Theory*, 46(2):388–404, 2000.

[54] B. Hajek and G. Sasaki. Link Scheduling in Polynomial Time. *IEEE Transactions on Information Theory*, 34(5):910–917, 1988.

[55] J. D. Hartline and T. Roughgarden. Optimal Mechanism Design and Money Burning. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC)*, 2008.

[56] 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, 1998.

[57] IEEE 802.15.2 Taskforce. Coexistence Mechanisms, 2008.

[58] K. Jain, J. Padhye, V. N. Padmanabhan, and L. Qiu. Impact of Interference on Multi-Hop Wireless Network Performance. In *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking (MOBICOM)*, pages 66–80, 2003.

[59] R. Jedermann, C. Behrens, D. Westphal, and W. Lang. Applying autonomous sensor systems in logisticsCombining sensor networks, RFIDs and software agents. *Sensors & Actuators: A. Physical*, 132(1):370–375, 2006.

[60] D. S. Johnson. Approximation Algorithms for Combinatorial Problems. In *Proceedings of the 5th Annual ACM Symposium on Theory of Computing (STOC)*, pages 38–49, 1973.

[61] A. R. Karlin, C. Kenyon, and D. Randall. Dynamic TCP Acknowledgement and Other Stories About $e/(e-1)$. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 502–509, 2001.

[62] R. M. Karp. Reducibility Among Combinatorial Problems. In *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[63] S. Khanna, S. Naor, and D. Raz. Control Message Aggregation in Group Communication Protocols. In *Proceedings of the 29th International Colloquium of Automata, Languagues and Programming (ICALP)*, pages 135–146, 2002.

[64] C.-Y. Koo, V. Bhandari, J. Katz, and N. H. Vaidya. Reliable Broadcast in Radio Networks: the Bounded Collision Case. In *Proceedings of the 25th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 258–264, 2006.

[65] P. Korteweg, A. Marchetti-Spaccamela, L. Stougie, and A. Vitaletti. Data aggregation in sensor networks: Balancing communication and delay costs. *Theoretical Computer Science*, 410(14):1346–1354, 2009.

[66] U. Kozat, I. Koutsopoulos, and L. Tassiulas. Cross-Layer Design for Power Efficiency and QoS Provisioning in Multi-Hop Wireless Networks. *Wireless Communications, IEEE Transactions on*, 5, 2006.

[67] B. Krishnamachari, D. Estrin, and S. Wicker. The Impact of Data Aggregation in Wireless Sensor Networks. In *Proceedings of the 22nd International Conference on Distributed Computing Systems Workshops (ICDCSW)*, pages 575–578, 2002.

[68] S. Krishnamoorthy, M. Robert, S. Srikanteswara, M. C. Valenti, C. R. Anderson, and J. H. Reed. Channel Frame Error Rate for Bluetooth in the Presence of Microwave Ovens. In *Proceedings of the Vehicular Technology Conference*, volume 2, 2002.

[69] S. Krumke, M. Marathe, and S. Ravi. Models and Approximation Algorithms for Channel Assignment in Radio Networks. *Wireless Networks*, 7(6):575–584, 2001.

[70] M. Kumar and S. Das. Pervasive Computing: Enabling Technologies and Challenges. *Handbook of Nature-Inspired and Innovative Computing: Integrating Classical Models with Emerging Technologies, Springer*, 2006.

[71] V. S. A. Kumar, M. V. Marathe, S. Parthasarathy, and A. Srinivasan. End-to-end packet-scheduling in Wireless Ad-Hoc Networks. In *Proceedings of the 15th annual ACM-SIAM symposium on Discrete Algorithms (SODA)*, pages 1021–1030, 2004.

[72] C. Law, A. Mehta, and K. Siu. A New Bluetooth Scatternet Formation Protocol. *Mobile Networks and Applications*, 8(5):485–498, 2003.

[73] Y. W. Law, L. van Hoesel, J. Doumen, P. Hartel, and P. Havinga. Energy-efficient Link-layer Jamming Attacks Against Wireless Sensor Network MAC Protocols. In *Proceedings of the 3rd ACM Workshop on Security of Ad hoc and Sensor Networks (SASN)*, pages 76–88, 2005.

[74] T.-H. Lee, J.-C. Lin, and Y. T. Su. Downlink Power Control Algorithms for Cellular Radio Systems. *IEEE Transactions on Vehicular Technology*, 44(1):89–94, 1995.

[75] K. Leung and L. Wang. Integrated Link Adaptation and Power Control for Wireless IP Networks, 2000.

[76] J. Li, C. Blake, D. De Couto, H. Lee, and R. Morris. Capacity of Ad Hoc Wireless Networks. In *Proceedings of the 7th ACM international conference on Mobile computing and networking (MOBICOM)*, pages 61–69, 2001.

[77] M. Li, I. Koutsopoulos, and R. Poovendran. Optimal Jamming Attacks and Network Defense Policies in Wireless Sensor Networks. In *Proceedings of the 26th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 1307–1315, 2007.

[78] L. Lovász. On the Ratio of Optimal Integral and Fractional Covers. *Discrete Mathematics*, 13:391–398, 1975.

[79] H. L. Mainak Chatterjee and S. K. Das. Rate Allocation and Admission Control for Differentiated Services in CDMA Data Networks. *IEEE Transactions on Mobile Computing*, 6(2):179–191, 2007.

[80] E. Maskin and T. Sjöström. *Handbook of Social Choice Theory and Welfare (Implementation Theory)*. North-Holland, Amsterdam, 2002.

[81] D. Monderer and M. Tennenholtz. k-Implementation. In *Proceedings of the 4th ACM Conference on Electronic Commerce (EC)* , pages 19–28, 2003.

[82] T. Moscibroda and S. Schmid. On Mechanism Design Without Payments for Throughput Maximization. In *Proceedings of the 18th IEEE Conference on Computer Communication (INFOCOM)*, 2009.

[83] T. Moscibroda, S. Schmid, and R. Wattenhofer. When Selfish meets Evil: Byzantine Players in a Virus Inoculation Game. In *Proceedings of the 25th annual ACM symposium on Principles of distributed computing (PODC)*, pages 35–44, 2006.

[84] T. Moscibroda and R. Wattenhofer. Coloring Unstructured Radio Networks. In *Proceedings of the 17th Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 39–48, 2005.

[85] T. Moscibroda and R. Wattenhofer. The Complexity of Connectivity in Wireless Networks. In *Proceedings of the 25th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2006.

[86] T. Moscibroda, R. Wattenhofer, and Y. Weber. Protocol Design Beyond Graph-based Models. In *Proceedings of the 5th ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets)*, 2006.

[87] T. Moscibroda, R. Wattenhofer, and A. Zollinger. Topology control meets SINR: the scheduling complexity of arbitrary topologies. In *Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing (MOBIHOC)*, pages 310–321, 2006.

[88] T. Nonner and A. Souza. Latency Constrained Aggregation in Chain Networks Admits a PTAS. In *Proceedings of the 5th International Conference on Algorithmic Aspects in Information and Management (ICAAIM)*, page 291, 2009.

[89] G. Noubir. On Connectivity in Ad Hoc Networks under Jamming Using Directional Antennas and Mobility. In *Proceedings of the Conference on Wired/Wireless Internet Communications (WWIC)*, pages 186–200, 2004.

[90] J. ORourke. Advanced problem 6369. *American Mathematical Monthly*, 88(10):769, 1981.

[91] C. H. Papadimitriou. Computational Aspects of Organization Theory. In *Proceedings of the European Symposium on Algorithms (ESA)*, 1996.

[92] C. H. Papadimitriou and E. Servan-Schreiber. The Origins of the Deadline: Optimizing Communication in Organizations. In *Complexity in Economics*, 1999.

[93] S. Papavassiliou and L. Tassiulas. Joint Optimal Channel Base Station and Power Assignment for Wireless Access. *IEEE/ACM Transactions on Networking*, 4(6):857–872, 1996.

[94] A. Pelc and D. Peleg. Feasibility and Complexity of Broadcasting with Random Transmission Failures. *Theoretical Computer Science*, 370(1-3):279–292, 2007.

[95] S. Pillai, T. Suel, and S. Cha. The Perron-Frobenius theorem: some of its applications. *IEEE Signal Processing Magazine*, 22(2):62–75, 2005.

[96] B. Radunovic and J.-Y. Le Boudec. Optimal Power Control, Scheduling, and Routing in UWB Networks. *Journal on Selected Areas in Communications*, 22(7):1252–1270, 2004.

[97] B. Radunovic and J.-Y. Le Boudec. Rate Performance Objectives of Multi-hop Wireless Networks. *IEEE Transactions on Mobile Computing*, pages 334–349, 2004.

[98] S. Ramanathan and E. L. Lloyd. Scheduling Algorithms for Multihop Radio Networks. *IEEE/ACM Transactions on Networking*, 1(2):166–177, 1993.

[99] L. G. Roberts. Aloha packet system with and without slots and capture. *SIGCOMM Computer Communication Review*, 5(2):28–42, 1975.

[100] T. Roughgarden. Stackelberg Scheduling Strategies. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 104–113, 2001.

[101] M. Satterthwaite. Strategy-proofness and Arrow's Condition: Existence and Correspondence Theorems for Voting Procedures and Social Welfare Functions. *Journal of Economic Theory*, 1975.

[102] G. Sharma, R. R. Mazumdar, and N. B. Shroff. On the Complexity of Scheduling in Wireless Networks. In *Proceedings of the 12th annual international conference on Mobile computing and networking (MOBICOM)*, pages 227–238, 2006.

[103] V. Shnayder, B. Chen, K. Lorincz, T. Jones, and M. Welsh. Sensor Networks for Medical Care. In *Proceedings of the 3rd international conference on Embedded networked sensor systems (SENSYS)*, volume 2, pages 314–314, 2005.

[104] N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri. Medians and Beyond: New Aggregation Techniques for Sensor Networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems (SENSYS)*, pages 239–249, 2004.

[105] S. Singh and C. S. Raghavendra. PAMAS - Power Aware Multi-Access Protocol with Signalling for Ad Hoc Networks. *SIGCOMM Computing and Communications Review*, 28(3):5–26, 1998.

[106] I. Solis and K. Obraczka. The Impact of Timing in Data Aggregation for Sensor Networks. In *Proceedings of the IEEE International Conference on Communications (ICC)*, volume 6, 2004.

[107] W. R. Stevens. *TCP/IP Illustrated, Vol.1: The Protocols*. Addison-Wesley, 1994.

[108] Y. C. Tay, K. Jamieson, and H. Balakrishnan. Collision-Minimizing CSMA and Its Applications to Wireless Sensor Networks. *IEEE Journal on Selected Areas in Communications*, 22(6):1048–1057, 2004.

[109] P. von Rickenbach and R. Wattenhofer. Gathering Correlated Data in Sensor Networks. In *Proceedings of the ACM Joint Workshop on Foundations of Mobile Computing (DIALM-POMC)*, pages 60–66, 2004.

[110] K. Wang, C. Chiasserini, R. Rao, and J. Proakis. A Joint Solution to Scheduling and Power Control for Multicasting in Wireless Ad Hoc Networks. *EURASIP Journal on Applied Signal Processing*, 2005.

[111] W. Wang, S. Eidenbenz, Y. Wang, and X.-Y. Li. OURS: Optimal Unicast Routing Systems in Non-Cooperative Wireless Networks. In *Proceedings of the 12th Annual International Conference on Mobile Computing and Networking (MOBICOM)*, pages 402–413, 2006.

[112] A. D. Wood, J. A. Stankovic, and G. Zhou. DEEJAM: Defeating Energy-Efficient Jamming in IEEE 802.15.4-based Wireless Networks. In *Proceedings of the 4th IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, pages 60–69, 2007.

[113] W. Xu, K. Ma, W. Trappe, and Y. Zhang. Jamming Sensor Networks: Attack and Defense Strategies. *IEEE Transactions on Networks*, 20(3):41–47, 2006.

[114] W. Xu, T. Wood, W. Trappe, and Y. Zhang. Channel Surfing and Spatial Retreats: Defenses against Wireless Denial of Service. In *Proceedings of the 3rd ACM Workshop on Wireless Security (WISE)*, pages 80–89, 2004.

[115] K. Yi and Q. Zhang. Multi-dimensional Online Tracking. In *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1098–1107, 2009.

[116] O. Younis and S. Fahmy. An Experimental Study of Routing and Data Aggregation in Sensor Networks. In *Proceedings of the IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pages 50–57, 2005.

[117] Y. Yu, B. Krishnamachari, and V. K. Prasanna. Energy-Latency Tradeoffs for Data Gathering in Wireless Sensor Networks. In *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communication Societies (INFOCOM)*, 2004.

[118] J. Zander. Distributed Cochannel Interference Control in Cellular Radio Systems. *IEEE Transactions on Vehicular Technology*, vol. 41(3):305–311, 1992.

[119] J. Zander. Performance of Optimum Transmitter Power Control in Cellular Radio Systems. *IEEE Transactions on Vehicular Technology*, 41:57–62, 1992.

[120] X. Zhou, S. Gandhi, S. Suri, and H. Zheng. eBay in the Sky: Strategy-Proof Wireless Spectrum Auctions. In *Proceedings of the 14th ACM international conference on Mobile computing and networking (MOBICOM)*, pages 2–13, 2008.

Publications

In the following, all my publications written as a student at ETH Zurich are listed.

1. On the Power of Uniform Power: Capacity of Wireless Networks with Bounded Resources. Chen Avin, Zvi Lotker, and Yvonne Anne Pignolet. In *Proc. 17th Annual European Symposium on Algorithms (ESA), Copenhagen, Danemark*, September 2009.

2. A Note on Uniform Power Connectivity in the SINR Model. Chen Avin, Francesco Pasquale, Zvi Lotker, and Yvonne Anne Pignolet. *Proc. 5th International Workshop on Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSOR), Rhodes, Greece*, July 2009.

3. Speed Dating despite Jammers. Dominic Meier, Yvonne Anne Pignolet, Stefan Schmid, and Roger Wattenhofer. In *Proc. 5th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS), Marina Del Rey, California, USA*, June 2009.

4. Tight Bounds for Delay-Sensitive Aggregation. Yvonne Anne Oswald, Stefan Schmid, and Roger Wattenhofer. In *Proc. 27th ACM Symposium on Principles of Distributed Computing (PODC), Toronto, Canada*, August 2008.

5. On the Windfall of Friendship: Inoculation Strategies on Social Networks. Dominic Meier, Yvonne Anne Oswald, Stefan Schmid, and Roger Wattenhofer. In *Proc. 9th ACM Conference on Electronic Commerce (EC), Chicago, Illinois, USA*, July 2008.

6. Word of Mouth: Rumor Dissemination in Social Networks. Jan Kostka, Yvonne Anne Oswald, and Roger Wattenhofer. In *Proc. 15th International Colloquium on Structural Information and Communication Complexity (SIROCCO), Villars-sur-Ollon, Switzerland*, June 2008.

7. What can be approximated locally? Christoph Lenzen, Yvonne Anne Oswald, and Roger Wattenhofer. In *Proc. 20th ACM Symposium on Parallelism in Algorithms and Architecture (SPAA), Munich, Germany*, June 2008.

8. Manipulation in Games. Raphael Eidenbenz, Yvonne Anne Oswald, Stefan Schmid, and Roger Wattenhofer. In *Proc. 18th International Symposium on Algorithms and Computation (ISAAC), Sendai, Japan, Springer LNCS 4835*, December 2007.

9. Complexity in Geometric SINR. Olga Goussevskaia, Yvonne Anne Oswald, and Roger Wattenhofer. In *Proc. ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC), Montreal, Canada*, September 2007.

10. Mechanism Design by Creditability. Raphael Eidenbenz, Yvonne Anne Oswald, Stefan Schmid, and Roger Wattenhofer. In *Proc. 1st International Conference on Combinatorial Optimization and Applications (COCOA), Xi'an, Shaanxi, China, Springer LNCS 4616*, August 2007.

11. How Optimal are Wireless Scheduling Protocols? Thomas Moscibroda, Yvonne Anne Oswald, and Roger Wattenhofer. In *Proc. 26th Annual IEEE Conference on Computer Communications (INFOCOM), Anchorage, Alaska, USA*, May 2007.

12. Luby-Rackoff Ciphers with Weak Round Functions. Ueli Maurer, Yvonne Anne Oswald, Krzysztof Pietrzak, and Johan Sjodin. In *Advances in Cryptology (EUROCRYPT), Lecture Notes in Computer Science, Springer-Verlag, vol. 4004, pp. 391-408,* May 2006.

13. Optimization of the MOVA Undeniable Signature Scheme. Jean Monnerat, Yvonne Anne Oswald, and Serge Vaudenay. In *Proc. Progress in Cryptology - Mycrypt 2005, LNCS vol. 3715, pp. 196-209, Springer-Verlag*, September 2005.