Mechanisms for Efficient Selfish Routing and Positioning in Ad Hoc Networks

A dissertation submitted to the
ETH Zurich
for the degree of Doctor of Technical Sciences

presented by
Luzius Ulrich Anderegg
Dipl. Informatik-Ingenieur ETH Zurich
born August 25, 1974, citizen of Zurich, Switzerland

accepted on the recommendation of
Prof. Dr. Peter Widmayer, ETH Zurich, examiner
Prof. Dr. Roger Wattenhofer, ETH Zurich, co-examiner
Dr. Leon Peeters, ETH Zurich, co-examiner

2006
Abstract

Wireless ad hoc networks constitute a great promise for the future, allowing anytime, anywhere communication without any central control. The underlying idea of wireless ad hoc networks is that devices equipped with a wireless transceiver, and located close to each other form a temporary network without the aid of existing infrastructure. Since the transmission range of each single device is restricted, most communication traffic has to go in multiple hops to its destination. The devices themselves are responsible for routing as well as for other network services. If the network is under control of a single entity, as for instance for rescue operations or military purposes, the devices cooperate out of pure interest, and the network services are guaranteed. However, if the devices are owned by different entities, as for instance for Internet access in an area with bad coverage, cooperation cannot be assumed anymore. In contrast, as the battery limits the lifetime of a device, it is reasonable to assume that a device tries to keep its contribution low while benefiting much from the network services of others.

Routing is a network service which is affected by the non-cooperation issue, as the devices need to be router for each other. Without any incentive, the forwarding of packets for other devices cannot be taken for granted. In this thesis, we present a routing protocol in which the source pays the devices on the path to the destination for forwarding its packets. The protocol is a variation of the well-known Vickrey-Clarke-Groves (VCG) mechanism, tailored to the setting of wireless ad hoc networks. Using game-theoretic concepts, we show that the protocol functions well with selfish devices, which rationally trade off incentives and costs. Further, the protocol finds efficient paths between devices communicating via multiple hops. Thus, the usage of energy is minimized, which is another critical issue in wireless ad hoc networks. We underline the practical applicability of the protocol by means of experiments.

The characteristics of the incentives in the routing protocol may influence the topology of the resulting network. As the payments in our VCG routing protocol are related to the planar positions of the devices,
the devices can be positioned such that the profit from the routing protocol is as large as possible. A first operation that influences the profit is the placement of additional devices. Given an existing network, a selfish agent owning a fixed number of additional devices, and given the set of communication requests in the near future, we want to find the positions for the additional devices such that the agent’s profit from the communication requests is maximum. We derive different computational complexity results for this operation. Further operations that influence the network topology are the deletion of devices, the movement of devices, or the change of the transmission range of devices. In the device deletion problem, we are to remove devices from a network such that the profit loss is minimum. The device movement and the device range change problem ask for the movement of devices respectively the change of the maximal transmission range of devices while maximizing the profit. The number of devices on which an operation can be performed, and the communication requests in the near future are given in all three operations. We present computational complexity results for several cases of the deletion, movement, and range change operations.
Zusammenfassung


Danksagung

Mein erster Dank gilt Peter Widmayer für die Unterstützung und Motivation während meiner Jahre in seiner Gruppe. Roger Wattenhofer und Leon Peeters danke ich für ihre Bereitschaft, sich als Korreferenten mit meiner Arbeit auseinanderzusetzen.

Ich bin froh, dass Mark, Jörg, Stephan, Alex, Michel, Beat, Barbara, Fabian, Tomas, Riko, Zsuzsa, Sonia, Marc, Leon, Paolo, Conrad, Franz, Konrad, Christoph, Gábor, David, Elias, Mirjam und Birgitta einen kurzen Abschnitt in ihrem Leben mit mir geteilt haben.

Seite Leer / Blank leaf
Contents

1 Introduction .................................................. 11
   1.1 Motivation ............................................. 11
   1.2 Notions from Complexity Theory .................. 14
   1.3 Notions from Mechanism Design .................. 16
   1.4 Summary of Results .................................. 19

2 An Incentive-Compatible and Efficient Routing Protocol for Selfish Ad Hoc Networks .......... 23
   2.1 Introduction .......................................... 23
       2.1.1 Related Work ................................... 24
       2.1.2 Model and Notation ............................. 26
       2.1.3 Summary of Results ............................. 32
   2.2 The Ad Hoc-VCG Protocols ........................... 32
       2.2.1 Basic Ad Hoc-VCG Protocol .................. 33
       2.2.2 Improved Ad Hoc-VCG Protocol ............... 37
   2.3 Analysis of the Ad Hoc-VCG Protocols ............... 41
       2.3.1 Incentive-Compatibility ....................... 41
       2.3.2 Efficiency ...................................... 46
       2.3.3 Overhead Message Complexity ................ 47
3 Strategic Placement of Devices

3.1 Introduction ................................................. 65
  3.1.1 Related Work ........................................ 66
  3.1.2 Model and Notation ................................. 68
  3.1.3 Summary of Results .............................. 70

3.2 Device Placement for Single Commodity ............... 71
  3.2.1 The Geometric Basis ............................... 71
  3.2.2 Single Device Placement for Single Commodity 76
  3.2.3 Multiple Identical Device Placement
       for Single Commodity ............................ 78

3.3 Single Device Placement for
       Multiple Commodities ............................ 80
  3.3.1 Single Maximization Diagram Approach ........ 83
  3.3.2 Multiple Maximization Diagrams Approach .... 86
  3.3.3 Extensions ........................................ 89

3.4 Multiple Identical Device Placement
       for Multiple Commodities ....................... 90

3.5 Individual Device Placement .......................... 102

3.6 Shortest Paths in Transmission Graphs .............. 105
  3.6.1 Sparse Distance Preserving Subgraphs ........ 105
  3.6.2 $\kappa$-hops Shortest Path ...................... 109

3.7 Conclusions and Open Problems ...................... 116
4 Strategic Deletion, Movement, and Range Change of Devices 117

4.1 Introduction .................................................. 117

4.1.1 Related Work ............................................. 118

4.1.2 Model and Notation ....................................... 119

4.1.3 Summary of Results ....................................... 120

4.2 Device Deletion ................................................ 122

4.2.1 Single Device Deletion for Single Commodity .......... 123

4.2.2 Single Device Deletion for Multiple Commodities 125

4.2.3 Multiple Device Deletion for Single Commodity ..... 126

4.2.4 Multiple Device Deletion for Multiple Commodities 126

4.2.5 Device Deletion in a Subset ............................ 133

4.3 Device Movement ............................................... 134

4.3.1 Single Device Movement for Single Commodity ..... 134

4.3.2 Single Device Movement for Multiple Commodities 135

4.3.3 Multiple Device Movement ............................... 136

4.3.4 Device Movement in a Subset ........................... 137

4.4 Transmission Range Change .................................. 137

4.4.1 Single Device Change for Single Commodity ......... 138

4.4.2 Single Device Change for Multiple Commodities .... 139

4.4.3 Multiple Device Change for Single Commodity ...... 140

4.4.4 Multiple Device Change for Multiple Commodities 141

4.4.5 Multiple Device Change for Single Commodity with Budget ............................................. 142

4.4.6 Transmission Range Change in a Subset .............. 146

4.5 Bundling Several Communication Requests ............... 146

4.6 Conclusions and Open Problems ........................... 148
Chapter 1

Introduction

1.1 Motivation

Traditional algorithm design typically assumes that the components of the system under consideration are obedient. In particular, each component follows the prescribed algorithm specification. Other concepts are around, such as systems with malicious or faulty components, but their share is small in comparison to the work with obedient components. In systems where the components belong to different entities with opposing goals, and where it is not possible to force the behavior of the components, the obedience assumption does not hold anymore. Such a component is also not malicious or faulty, but tries to attain its own goals, and deviates from the prescribed algorithm specification if this is in its own benefit. The assumption that a component is selfish (or rational) makes more sense in such systems, and motivates the design of algorithms in which the selfish behavior of each component coincides with following the algorithm specification.

One possibility to ensure that a selfish component acts according to the algorithm specification is to provide some incentive for "correct" behavior.
Since a long time, the field of economics has studied incentive systems, that is, how much has to be paid to whom at which point in time. The main tools in economics to study incentive systems are game theory and its subfield mechanism design. Game theory aims at understanding situations where so-called selfish agents interact, and mechanism design deals with the setup of rules such that the induced game ends in a desired situation. However, economists mainly focused on the existence of incentive systems, and less on computational aspects, such as the running time or the communication complexity arising while implementing the incentive system in a computerized setting. The emergence of the Internet, which is owned, operated, and used by different independent entities, each acting selfishly, has created many settings where algorithmic issues play an important role. This new type of setting where incentive issues and algorithmic issues are addressed together has come to be known as *algorithmic mechanism design* [64].

The algorithmic mechanism design approach is relevant in several problems in the Internet such as web caching, distributed task allocation, peer-to-peer file sharing, or the design of effective protocols for different layers in the Internet protocol stack [34, 33, 75]. One of the most investigated problems is the combinatorial auction (see [23] for a whole textbook dedicated to combinatorial auctions). In a combinatorial auction, several goods are auctioned together and the participants are allowed to bid on arbitrary combinations of goods. The complexity as well as the often distributed setting of combinatorial auctions make the use of computers indispensable. The applications of combinatorial auctions nowadays vary from the allocation of mobile phone spectrum rights to airport slot allocations and procurement auctions in e-commerce. The application range seems to grow further in the future, with potential applications in transportation [5, 4, 40] and new business models arising in the Internet [58].

Besides the Internet, wireless ad hoc networks form another distributed system with independent selfish agents. In an ad hoc network, wireless devices cannot rely on a fixed infrastructure accessible through access points. Instead, they have to establish a network among themselves by communicating via radio. Due to the restricted transmission range of each single
1.1. Motivation

device, it is typically not possible to establish a direct communication between every pair of devices. Hence, if two devices are not within their mutual transmission range, the communication has to take place via some intermediate devices from the source to the destination. Hybrid networks have similar characteristics as wireless ad hoc networks, with a fixed infrastructure on a limited number of access points. Therefore, some wireless devices cannot directly reach an access point, and to extend the coverage of the network, a wireless device can use other devices as relay devices until an access point is reached. The lack of infrastructure has a further consequence for the wireless devices, as they are in general battery operated, which limits the lifetime of the device to a few hours.

The aforementioned characteristics of wireless ad hoc networks make them an exemplar of algorithmic mechanism design. Indeed, selfish issues arise on different layers of the protocol stack. On the medium access layer, a device tries to obtain an unfair share of access to the channel, and this action decreases the share of other devices. On the network layer, forwarding packets drains the battery, and decreases the lifetime of a device. Therefore, a device is not willing to route packets for other devices without being compensated for its expenses. On the transport layer, a device can implement its own congestion control algorithm in order to increase the throughput at the cost of other devices. Various other selfish issues in wireless ad hoc networks are described in the survey article [75].

In this thesis, we investigate the problem of routing in wireless ad hoc networks with selfish agents. The challenging question in the routing problem is how the selfish devices can be convinced to forward data packets for other devices. If the selfish devices do not receive an incentive for doing so, then they cannot be expected to forward packets, and essentially no communication is taking place in the network. Our mathematical abstraction to study the routing problem is a graph. The devices in the network translate to vertices in the graph, direct connections between two devices to edges, and the cost of an edge corresponds to the transmission cost between the two devices. Since the transmission cost is closely related to the distance between the two devices, the underlying geometry becomes important. We assume that the devices lie in the plane, and that the transmission cost is
proportional to a power of the Euclidean distance.

The first part of the thesis considers the development of a routing protocol for wireless ad hoc networks. Our tools to analyze the protocol stem from game theory. Due to the peculiarities of wireless ad hoc networks, we investigate a distributed setting, which differs from the centralized setting mainly considered in traditional mechanism design. We propose a protocol which implements a distributed version of the classical Vickrey-Clarke-Groves (VCG) mechanism.

In the second part of the thesis, we slightly change perspective. Given a known protocol, an entity controlling some device is faced with the question how it can exploit the protocol. For the VCG payments of our routing protocol, this gives rise to the question how to position devices in the network so as to maximize the resulting profit. We study this kind of question for an entity newly joining a network, and ask for the placement with largest profit. Next, we extend this question to further operations. An entity controlling devices in the network can move devices, or increase the transmission range of devices, or be forced to remove devices from the network. The goal is to maximize the profit respectively to minimize the profit loss under such operations. This change of perspective also causes a change of focus to combinatorial and geometric problem aspects.

In the remainder of this chapter, we first introduce some notions from complexity theory in Section 1.2. Next, the notions from game theory required in this thesis are described in Section 1.3. Section 1.4 summarizes the results of the thesis.

1.2 Notions from Complexity Theory

This section gives a short overview of the notions from complexity theory used in this thesis. A more extensive treatment of these notions can be found in any textbook on complexity theory, as for instance [9] and [67].

As a measure for the computational complexity, we consider the time an algorithm needs to produce a solution. This measure naturally depends
1.2. Notions from Complexity Theory

on the underlying machine model. All propositions about running times of algorithms refer to a model with algorithms written in a PASCAL-like language and run on a real machine. Using this model, the running time of an algorithm is equal to the number of instructions it executes. As is standard in complexity theory, we make use of the $O$-notation to express asymptotic worst-case running time.

We are concerned with the complexity classes $\mathcal{P}$ and $\mathcal{NP}$, which are defined with respect to a decision problem. A decision problem has a set of input instances, partitioned into YES-instances and NO-instances, and the problem asks, for any instance, whether it is a YES-instance. The class $\mathcal{P}$ is defined as the set of decision problems for which there exists a deterministic algorithm which decides, for any instance and in time bounded by a polynomial of the input size, whether the instance is a YES-instance. The class $\mathcal{NP}$ is defined as the set of decision problems for which there exists a nondeterministic algorithm which decides, for any instance and in time bounded by a polynomial of the input size, whether the instance is a YES-instance. For many important decision problems we neither know of a deterministic polynomial-time algorithm, nor can prove that no deterministic polynomial-time algorithm exists. Hence, relations of the form "problem $x$ is at least as hard as problem $y$" between decision problems have been established. To this end, one can use a polynomial-time reduction. A decision problem $\Pi_1$ is polynomial-time reducible to a decision problem $\Pi_2$ if there exists an algorithm transforming, in time bounded by a polynomial of the input size, any instance of $\Pi_1$ into an instance of $\Pi_2$, such that the instance of $\Pi_1$ is a YES-instance if and only if the transformed instance of $\Pi_2$ is a YES-instance.

Polynomial-time reductions allow a further classification of problems. A decision problem $\Pi$ is $\mathcal{NP}$-hard if there exists a polynomial time reduction from any problem in $\mathcal{NP}$ to $\Pi$. If, in addition, the problem $\Pi$ is in the class $\mathcal{NP}$, then the problem is said to be $\mathcal{NP}$-complete. From a current pragmatic point of view, $\mathcal{NP}$-complete problems are considered as problems that cannot be solved efficiently, whereas problems in $\mathcal{P}$ are efficiently solvable. Due to the transitivity of polynomial-time reductions, it suffices to find a polynomial-time reduction from one $\mathcal{NP}$-hard problem
in order to prove $\mathcal{NP}$-hardness of a problem.

In practical applications, optimization problems are more common than decision problems. In an optimization problem, we are to find an optimum solution with respect to a given cost or profit measure for any input instance.

### 1.3 Notions from Mechanism Design

Similar to complexity theory in the previous section, we introduce here some notions from mechanism design needed in the thesis. Mechanism design, also known as implementation theory, is a subfield of game theory. Several textbooks provide a sound general introduction to game theory. Osborne and Rubinstein [66] and Osborne [65] provide a good reference for game theory, and Chapter 23 in Mas-Colell et. al [57] gives an overview on mechanism design. More computer science-oriented overviews can be found in the seminal paper on algorithmic mechanism design by Nisan and Ronen [64] or in the thesis of Parkes [69].

Broadly speaking, a mechanism design problem is described as follows. The goal is to solve an instance of some optimization problem, without knowing all input data of the instance. Instead, several selfish entities each know some portion of the input data. An algorithm is allowed to ask each entity for its portion of knowledge. However, when the algorithm asks, the answer of the entity is not necessarily the true input data, but may be any data that the entity finds suitable. Our task is to design a mechanism with payments such that reporting the true data is the best thing an entity can do, and moreover, such that the instance of the optimization problem we are interested in is solved.

In the terminology of mechanism design, an entity is called an agent, and the private data of an agent is called the type of an agent. The type encapsulates the relevant information about an agent needed for the optimization problem. Further, an agent evaluates each feasible outcome of the optimization problem according to its utility function. In this thesis, the utility function of an agent is equal to the payment the agent receives from
the mechanism minus the cost incurred by the agent in the selected outcome, and each agent acts in such a way that its utility is maximized. This form of utility function is called quasi-linear. We also use the term profit as a synonym for utility. A mechanism is called incentive-compatible if each agent maximizes its utility by truthfully reporting its type, assuming that all other agents report truthfully. A mechanism is incentive-compatible in dominant strategies if each agent maximizes its utility by truthfully reporting its type, regardless of the reports of the other agents.

The probably best known positive result in mechanism design is the class of Vickrey-Clarke-Groves (VCG) mechanisms [82, 20, 43]. Before presenting the VCG mechanism in its general form, we first illustrate its usage for finding a shortest path in a graph. The setting consists of a graph in which each edge is an agent with quasi-linear utility function. Each edge has an associated cost for using it, which is only known to the agent, that is, the cost is the type of the agent. The optimization problem consists in finding a shortest path with respect to the edge costs between two given distinct vertices. Let us first consider the straightforward mechanism where the mechanism first asks each agent about its cost, and subsequently selects a shortest path based on the reported costs of all agents. The mechanism pays each agent participating in the selected shortest path the reported costs, and all other edges receive a payment of zero. With this mechanism, if an agent reports its true cost and is part of the shortest path, then its utility is equal to zero. If such an agent reports cost slightly more than its true cost, and still is part of the shortest path, then its payment is also slightly larger than the true cost, and the agent has a positive utility. However, if the agent exaggerates too much with its cost, then it risks not being part of the shortest path, yielding zero utility. Even though the exact behavior of an agent is difficult to predict, the mechanism is certainly not incentive-compatible.

The VCG idea to make the mechanism incentive-compatible is to pay an agent more than it has reported. The other steps in the mechanism remain the same. The mechanism asks each agent about its cost, and selects a shortest path based on all reported costs. The payments are set as follows. An agent participating in the shortest path receives a payment equal to the
difference of the total cost of a shortest path between the two distinct vertices in the graph when the edge is taken out of the graph (or, equivalently, with the cost of the edge set to infinity) and the total cost of a shortest path in the graph when the cost of the agent is assumed to be zero. An agent not participating in the shortest path still receives nothing. For this mechanism, it can be shown that each agent maximizes its utility by reporting the true edge cost. Thus, the mechanism is incentive-compatible.

We continue with the general VCG mechanism. Let \( \mathcal{O} \) be the set of feasible outcomes of the optimization problem under study, and \( c_i(o) \) the true cost of agent \( i \) when outcome \( o \in \mathcal{O} \) is selected. We further assume that the optimization problem is a minimization problem. The mechanism asks each agent about its cost in each possible outcome. We denote the reported cost of agent \( i \) for outcome \( o \) by \( \hat{c}_i(o) \). Note that in the shortest path example, the set of outcomes is the set of possible paths between the two distinct vertices. Since the cost for each agent only depends on whether it is part of the shortest path or not, and not on the complete shortest path, it only has to report its cost for the outcome when it is part of the shortest path. Further, the cost for an outcome in which it is not participating in the shortest path is equal to zero. The VCG mechanism selects the outcome \( o^* \in \mathcal{O} \) according to

\[
  o^* = \arg \min_{o \in \mathcal{O}} \sum_{j \text{ agents}} \hat{c}_j(o),
\]

and pays agent \( i \) an amount of

\[
  h_i(\hat{c}^{-i}) = \sum_{j \neq i} \hat{c}_j(o^*)
\]

where \( \hat{c}^{-i} \) is the vector of reported cost of all agents except agent \( i \). The term \( h_i(\cdot) \) is an arbitrary function dependent on \( \hat{c}^{-i} \), and consequently independent of \( \hat{c}_i \). The most common choice for \( h_i(\cdot) \) is \( \sum_{j \neq i} \hat{c}_j(o^*_{-i}) \) where \( o^*_{-i} \in \mathcal{O} \) is the outcome according to

\[
  o^*_{-i} = \arg \min_{o \in \mathcal{O}} \sum_{j \neq i} \hat{c}_j(o),
\]
that is, the optimal outcome if $i$ were not present. A VCG mechanism is incentive-compatible, even in dominant strategies. If, moreover, the agents are selfish with quasi-linear utility functions, then a VCG mechanism exactly solves optimization problems of the form $\min \sum_{\text{agents}} c_i(o)$ respectively $\max \sum_{\text{agents}} c_i(o)$.

1.4 Summary of Results

In Chapter 2, we present routing protocols for wireless ad-hoc networks with selfish agents. After a description of the requirements for such a protocol in Section 2.1, we propose two routing protocols, the basic ad hoc-VCG protocol and the improved ad hoc-VCG protocol, in Section 2.2. Next, we show in Section 2.3 that the basic ad hoc-VCG protocol fulfills the requirements of cost-efficiency and incentive-compatibility. In the same section, we prove that the improved ad hoc-VCG protocol shares the properties of the basic version of the protocol, and in addition has respectable overhead message complexity. In order to obtain incentive-compatibility, the source has to pay the devices that forward the data packets. Hence, the total cost that the source faces equals the sum of these payments and the cost to transmit the data packets to the first intermediate device. This raises our interest in the overcharge ratio, the ratio between the total cost for the source and the actual cost faced by the devices that forward the data packets. We provide an upper bound on the overcharge ratio in Section 2.3. We conclude the chapter in Section 2.4 with experiments on the performance of the two protocols, showing feasibility of both protocols. We also experimentally evaluate the average overcharge ratio.

Chapter 3 deals with a problem that arises from taking the viewpoint of a single agent. An agent that knows the protocol and the corresponding payments, as well as the device pairs wishing to communicate, can exploit this knowledge. The payments in our routing protocols are inherently connected to the position of the devices in the plane. Hence, an agent that owns a set of devices, and newly enters an existing wireless network, can choose the positions for its devices such that its utility is maximal. Sec-
tion 3.1 introduces this problem. Next, we start our investigations with the easiest case of a single commodity, that is, a single device pair wishing to communicate, where an agent possesses a single device. In Section 3.2, we propose a polynomial-time algorithm using a layer graph for this case. The layer graph encodes the use of the additional single device, and transforms the problem into a standard shortest path search. In the same section, we also present a polynomial-time algorithm for the case with multiple identical devices for a single commodity, using an extension of the layer graph approach. We proceed in Section 3.3 with the more difficult case of a single device for multiple commodities. This case is solved via a subdivision of the plane into cells in which the optimal position can be determined in polynomial time. As the number of cells is polynomially bounded as well, the running time for the algorithm is polynomial. In Section 3.4, we show that the case with multiple identical devices for multiple commodities is \( \mathcal{NP} \)-hard. To this end, we construct a reduction from planar \textsc{Exact Cover by 3-Sets} where each element appears in either two or three sets. We continue with a \( \mathcal{NP} \)-hardness proof for the case with multiple individual devices and multiple commodities in Section 3.5, showing that this case already becomes \( \mathcal{NP} \)-hard for two commodities by a reduction from \textsc{Partition}. We conclude the chapter in Section 3.6 with investigations on the pure shortest path problem and the hop-restricted shortest path problem in transmission graphs. The section aims to identify possible improvements for the algorithms from the previous sections.

After considering the problem of placing additional devices, a natural next step is to consider other strategic device operations. In Chapter 4, we investigate three further strategic device operations. In strategic device deletion, an agent is forced to remove a fixed number of devices and aims to do this such that its utility loss is minimized. In strategic device movement, an agent can choose new positions for its devices with the goal to maximize its utility, whereas strategic transmission range change allows an agent to change the maximal transmission range of its devices to maximize the profit. After an introduction in Section 4.1, we start with strategic device deletion in Section 4.2. We propose a polynomial-time algorithm for the case where either a single device or a single commodity is involved.
For the general case with multiple devices and multiple commodities, we prove \( \mathcal{NP} \)-hardness. To this end, we use a similar reduction as in the \( \mathcal{NP} \)-hardness proof for the multiple identical device placement problem for multiple commodities in Section 3.4. We continue with strategic device movement in Section 4.3. We show that we can solve the single device cases in polynomial time, and that the case of multiple devices for multiple commodities is \( \mathcal{NP} \)-hard. Here, the polynomial-time algorithms use the straightforward approach of exhaustively trying all possibilities. The \( \mathcal{NP} \)-hardness proof goes along the lines of the one for the multiple individual device placement problem for a single commodity in Section 3.5. The case with multiple devices for a single commodity remains open. As a last strategic operation, we consider transmission range change in Section 4.4. From a complexity theoretical point of view, this situation is similar to the identical device placement operation. The cases with either a single device or a single commodity are polynomial-time solvable, whereas the general case with multiple devices and multiple commodities is \( \mathcal{NP} \)-hard. To obtain these results, similar techniques are used as for the other operations. Next, we consider a variation of the range change operation, in which the total increase of the maximal transmission ranges is bounded by a budget, and the number of devices to be changed is not restricted. We prove this problem to be \( \mathcal{NP} \)-hard by a reduction from SUBSET SUM. Section 4.5 ends the chapter with some investigations on a different cost model. In this cost model, the cost of an edge is charged exactly once if communication takes place over this edge, independent of the number of source-destination pairs participating in the communication over the edge.
Seite Leer / Blank leaf
Chapter 2

An Incentive-Compatible and Efficient Routing Protocol for Selfish Ad Hoc Networks

2.1 Introduction

Routing is one of the key tasks in networks. It is responsible to determine a path for the data packets from a source to a destination. The most simple variant of routing, unicast routing, determines a route between a single source and a single destination. There are further variants of routing for other communication patterns as for instance multicast routing, where routes between a single source and multiple destinations have to be found. In this thesis, we concentrate on unicast routing, and use the term routing for unicast routing when no ambiguity arises.

Many routing protocols for wireless ad hoc networks assume that all
devices in the ad hoc network are cooperative, in particular, that they are willing to act as intermediate devices in a routing path by forwarding data for other network devices. This cooperativeness assumption may be reasonable in some settings. But willingness to cooperate can certainly not be assumed generally in an ad hoc setting since forwarding data for other network devices can drain the battery of a device, without this device ever being the source or the destination of the data that it forwards. Moreover, a device reduces the bandwidth available for itself by using bandwidth for the data transfers of other devices. If the devices in a network are not owned by a single entity, but by profit-oriented independent entities, then the devices are indeed selfish. Thereby, routing in wireless ad hoc networks constitutes an exemplar of mechanism design in computer science. In this chapter, we propose a method of coping with this selfishness. Moreover, the method yields other important properties of a routing protocol. The protocol yields an efficient routing path, that is, the scarce battery resource of the devices is not charged too much. Also, the protocol is a reactive routing protocol, constructing a route on-demand whenever a source device wants to deliver some data packets to a destination device.

2.1.1 Related Work

Routing in ad hoc networks without selfish devices has been the subject of intense research over the past few years, and has resulted in numerous proposals for routing protocols (see [80] for a survey). In general, routing protocols can be classified as proactive and reactive routing protocols. In a proactive routing protocol, each device stores routes in a table, and looks up in this table when required. Consequently, whenever a change occurs in the network, the tables have to be updated as well. The most prominent representatives of proactive routing are the wireless routing protocol (WRP) [62] and the destination sequenced distance vector (DSDV) [70]. On the other hand, a reactive routing protocol only determines a route when a source device initiates a request to send data traffic. Dynamic source routing (DSR) [15], ad hoc on-demand distance vector routing (AODV) [24], and the temporally ordered routing algorithm (TORA) [68] form the most
known protocols in this class. Hybrid approaches with components from proactive protocols as well as components from reactive protocols also exist. General statements about routing protocols from an algorithmic point of view are rare, and most analyses are based on evaluation by simulation. The exception are some specific types of ad hoc networks as for instance ad hoc networks where the device positions are known [81, 88].

Selfishness within wireless network routing has been studied only recently. Most approaches fall into one of two categories: approaches rewarding cooperative devices and approaches punishing non-cooperative devices. In the first category, devices forwarding data packets receive monetary incentives for their service. In the work of Buttyán and Hubaux [17], the sender (or the destination) pays intermediate devices using a virtual currency called NUGLETS, for which different payment models are proposed. In a follow-up work [18], the same authors enhance their results, introducing a model with a credit counter in each device, and analyzing four rules when a device forwards data packets for other devices. Zhong et al. [86] presented the SPRITE protocol, which contains a payment scheme such that every device acts truthfully in a game-theoretic sense. Based on the ideas presented in this chapter, Zhong et al. [87] extended the protocol to work when the transmission between two devices is only successful with a certain probability. In a series of papers [84, 83, 85], Wang et al. proposed a unicast routing protocol as well as routing protocols for other communication patterns. For multi-hop cellular networks, Jakobsson et al. [47] uses micro payments to stimulate cooperation.

In the second category, non-cooperative devices are identified based on a reputation system and circumvented in the routing process. Marti et al. [56], Buchegger and Le Boudec [16], and Michiardi and Molva [59] propose different repudiation systems with respect to how this information is propagated through the network. A survey on selfish behavior in all layers of the protocol stack in wireless ad hoc networks can be found in [75].
2.1.2 Model and Notation

A wireless ad hoc network is modeled by an ad hoc graph \( A = (V, E, g) \) consisting of a vertex set \( V = \{1, \ldots, n\} \), a set \( E \subseteq V \times V \) of directed edges, and a cost function \( g : E \rightarrow \mathbb{R} \). The set \( V \) models the wireless devices embedded in the plane, each with a unique identification. We will denote the identification of device \( v \) by \( id_v \). For simplicity, we sometimes use the terms vertex and device interchangeably. There exists a directed edge \((u, v) \in E\) if device \( v \) is inside the transmission range of device \( u \). We assume that the resulting graph is 2-connected, that is, the removal of a single vertex does not disconnect the graph into two components.\footnote{This assumption is standard in selfish settings. An intuitive explanation why 2-connectedness is necessary is that a bottleneck vertex (i.e., a vertex whose removal disconnects the graph) can demand an arbitrary high payment for forwarding a packet.}

Modern wireless devices can alter the emission power level for transmitting a packet up to a maximal level, and can consequently vary their transmission range. As we assume omnidirectional antennas, a device cannot control the direction in which it sends data, and thus data are broadcast to all devices inside the transmission range. The power \( P_{u,v}^{\text{rec}} \) at which a device \( v \) receives a signal emitted by \( u \) with emission power \( P_u^{\text{emit}} \) is \cite{29}:

\[
P_{u,v}^{\text{rec}} = \frac{\Gamma}{|uv|^\alpha} P_u^{\text{emit}},
\]

where \(|uv|\) is the Euclidean distance between devices \( u \) and \( v \), \( \alpha \) is the path loss exponent, and \( \Gamma \) is a constant. The path loss exponent depends on the environment conditions and varies between one and six. If the receiving power \( P_{u,v}^{\text{rec}} \) exceeds a minimum receiving power threshold \( P_{\text{min}}^{\text{rec}} \), device \( v \) can successfully receive the packet. Ideally, a device uses exactly the emission power such that the receiving power equals \( P_{\text{min}}^{\text{rec}} \). We denote by \( P_{u,v}^{\text{min}} \) the minimum emission power that device \( u \) must use to just reach device \( v \).

Unfortunately, since a device does not know the positions of the other devices, it does not know \( P_{u,v}^{\text{min}} \) either. However, device \( u \) can determine
the minimum emission power $P_{u,v}^{\text{min}}$ from itself to a device $v$ with the help of a so-called loopback technique. The loopback technique works as follows. Device $u$ sends a test packet using emission power $P_u^{\text{emit}}$. Upon receiving this test packet, device $v$ determines the power $P_{u,v}^{\text{rec}}$ at which it receives the packet, and sends this value back to device $u$. Given the values $P_u^{\text{emit}}$ and $P_{u,v}^{\text{rec}}$, as well as the minimum receiving power threshold $P_{\text{min}}^{\text{rec}}$, device $u$ can now compute $P_{u,v}^{\text{min}}$ using Equation 2.1.1 as:

$$P_{u,v}^{\text{min}} = \frac{P_u^{\text{emit}} \cdot P_{\text{min}}^{\text{rec}}}{P_{u,v}^{\text{rec}}}.$$ 

A cost-of-energy value $e_u$ models the inconvenience caused to the device $u$ by asking it to forward a data packet. This value may depend on the current battery level of the device, on its cost for recharging the battery, and on other internal parameters. Moreover, the cost-of-energy value is known only to the device itself.

The cost function $g : E \rightarrow \mathbb{R}$ models the cost of transmitting a unit-size data packet from device $u$ to device $v$ along the edge $(u, v)$. We set the transmission cost equal to the emission power needed at $u$ to reach $v$ weighted by the private cost-of-energy value of $u$, that is,

$$g(u, v) = e_u \cdot P_{u,v}^{\text{min}}.$$ 

We ignore other types of energy consumption, such as listening to signals, as they tend to be magnitudes smaller than the emission power.

A path in $A$ is a sequence of devices in which any two consecutive devices have a common edge. An $s-t$-path is a path where $s$ is the first device and $t$ is the last device in the sequence. The cost of an $s-t$-path $\sigma(s, t) = \{s = \sigma(1), \sigma(2), \ldots, \sigma(l(\sigma)) = t\}$, where $l(\sigma)$ is the number of devices on the path, is the sum of the transmission costs along the path, i.e. it is equal to $\sum_{i=1}^{l(\sigma)-1} g(\sigma(i), \sigma(i + 1))$. A shortest path between two devices $s$ and $t$ is a path with minimum cost. We use $SP_A(s, t)$ to denote the shortest path between $s$ and $t$ in the ad hoc graph $A$. Further, let

---

3 Similar techniques for power control are widely used in cellular networks [73].
\(|SP_A(s, t)|\) be the cost of the shortest path \(SP_A(s, t)\). Mostly, the underlying graph \(A\) is clear, and we use \(SP(s, t)\) as a shorthand for \(SP_A(s, t)\). A shortest replacement path for a device \(v\) is a path with minimum cost that does not contain device \(v\). Similarly to before, let \(SP_{A-v}(s, t)\) be the shortest replacement path for \(v\) between \(s\) and \(t\) in the ad hoc graph \(A\), \(|SP_{A-v}(s, t)|\) its total cost, and \(SP^{-v}(s, t)\) a shorthand for \(SP_{A-v}(s, t)\).

As each device acts selfishly, it only forwards a data packet if it is reimbursed with an appropriate amount of money. Specifically, each device has a utility for each possible path, and it wants to maximize this utility. In our case of routing, the utility \(U_v\) of a device \(v\) is:

\[
U_v = pay_v - cost_v
\]

or, in words, the utility of a device is equal to the payment the device receives minus the cost it incurs. The cost \(cost_v\) for an intermediate device on the communication path is the cost for forwarding a data packet, which we assume to be equal to the cost of the edge along which the data packet travels. As outlined above, the forwarding cost for a device \(u\) is composed of its private cost \(e_u\), and of its minimum emission power \(P_{u,v}^{\text{min}}\), computable by \(u\) together with its neighbor \(v\). The product \(g(u, v) = e_u \cdot P_{u,v}^{\text{min}}\) forms the type of device \(u\). Since a device is selfish, it may report false information about its type when asked, and will indeed do so if this yields a higher utility. When acting as the source device, we assume that a device has a high value from sending a data packet to the destination. This means that a source device is willing to pay each intermediate device for each data packet that it sends to the destination device, and that it yields a positive utility regardless of these payments.\(^4\) The utility of the destination device is a constant if receiving as we assume that it is in its own interest to receive the data packets.

Since wireless ad hoc networks have no central control, the setting is inherently distributed. In most classical mechanism design settings, there is a

\(^4\)This assumption is quite strong as it essentially requires a device to act non-selfishly, whenever it is a source. However, there is a tradition in mechanism design to treat the source (or auctioning agent) in such a special way (see Chapter 23.C, pp. 880 - 881 in [57]). See [31] for a mechanism which is based on the mechanism presented here, and allows the source to act selfishly.
center to which each device can send some information, which then makes a decision, and sends the decision back to the devices. In contrast, in our setting, any device can only send packets to devices inside its transmission range. The devices themselves are responsible to compute intermediary results and to forward information on behalf of other devices. As devices are selfish, they possibly compute incorrect intermediary results or drop respectively alter packets intended to be forwarded if this increases their utility. Thus, one must design a distributed mechanism to determine the path of devices along which the packets are sent, and the payment for each device on this path. A protocol is the implementation of such a distributed mechanism, for which we require the following properties.

**Incentive-compatibility**

A mechanism is incentive-compatible in classical mechanism design if each selfish agent truthfully reports its type, assuming that all other agents truthfully report their type. As indicated in the last paragraph, there are issues in the wireless ad hoc network setting such as intentional dropping of packets in addition to truthfully reporting, due to the distributed nature of the setting. To take these issues into account, we extend the definition of incentive-compatibility for a protocol. We say that a protocol is incentive-compatible if each agent maximizes its utility by following the protocol specification, assuming that all other agents follow the specification. In our setting, this means that any device $v$ reports its true cost-of-energy value $e_v$, its true emission power $P_{v}^{emit}$, and the true minimum emission power $P_{u,v}^{min}$ for all devices $u \neq v$, and that it does not alter or drop packets.\(^5\)

Incentive-compatibility is useful from the viewpoint of the protocol as well as from the viewpoint of a single device. It is desirable for the protocol because the problem under study is difficult to solve if we do not even know the input (see efficiency property listed below). It is desirable for a single

---

\(^5\)The concept of incentive-compatibility in dominant strategy from mechanism design is a very strong solution concept, and difficult to achieve in a distributed setting as we have it here. Intuitively, the reason for that lies in the fact that part of the type of a device is determined by the neighbors of the device.
device because it keeps the device from wasting efforts in reasoning about whether it should follow the protocol.

**Efficiency**

Energy-efficiency is a key objective in many routing protocols (see [55] for a survey and [29, 13] for more recent work). It represents the traditional approach of dealing with the problem of battery drainage from a global point of view. The total energy of a path is the sum of the emission powers needed at the source and at each intermediate device on the path. An energy-efficient routing protocol ensures that a data packet from a source device is routed to a destination device along a path with minimum total energy, and thereby, energy consumption in the whole network is minimized. Energy-efficiency does not deal with the problem of battery drainage from the point of view of a single device. If a device with low battery level lies on the energy-efficient path, then energy-efficiency does not take this into consideration. The cost-of-energy value incorporates the viewpoint of a single device. Hence, we are looking for the most cost-efficient path, that is, the shortest path with respect to the edge cost $g(u, v)$.

**Reactive nature**

As holding lots of routing information at each device is too expensive, we want the protocol to be reactive. A reactive routing protocol only takes action and starts computing a path when a device initiates a communication request. Consequently, such a protocol consists of two phases: a route discovery phase and a data transmission phase. In the route discovery phase, control packets are sent around to determine a source-destination path, including the corresponding payments that are to be made to the intermediate devices for each forwarded data packet. In the data transmission phase, the source sends data packets along the computed path to the destination. We assume that every communication request consists of a relatively large number of data packets. This motivates our next assumption that devices are willing to forward control packets for free because of a potentially large
utility in the data transmission phase. In fact, we assume that all devices participate in the route discovery phase without receiving payment for it. A reactive routing protocol also has a route recovery phase which is needed when the computed path breaks. The route recovery phase is not considered in this thesis.

Overhead message complexity

The overhead message complexity measures the number of control packets exchanged during the route discovery phase. Although the number of control packets for each individual device is negligible, the total number of control packets over all devices should be small to avoid congestion in the network. Hence, we require that the overhead message complexity is small as this allows the protocol to work in networks of reasonable size.

Further remarks

We close this section with some further remarks about the model. VCG mechanisms for shortest path problems in other settings have been criticized for overcharging. In a worst case scenario, the sum of payments along a path is much larger than the actual cost of the path [8]. We investigate this issue of overcharging in Section 2.3.4 from a theoretical point of view, and in Section 2.4.2 from an experimental point of view.

We do not focus on payment delivery, but only on payment computation. Money transfers between devices are assumed to be carried out by means of a central bank such as in [86], or by tamper-proof hardware which is included in each device to store the money, such as in [17]. The money transfer does not take place until the destination has confirmed that it has received the data packets. Consequently, intermediate devices do not receive any payment if the data transmission phase fails for any reason.

Some steps in the protocol specification require cryptographic techniques (i.e. digital signatures). We assume that these techniques are provided by a standard cryptographic solution.
Because of the complexity of the problem, we only consider the static scenario here. The devices do not move around in a static scenario from the point in time when a device initiates a communication request until all data packets are transferred from the source to the destination.

To summarize, the routing problem involves an ad hoc graph $A = (V, E, g)$, and a source $s \in V$ which wants to send data packets to a destination $t \in V$. The goal is to design an incentive-compatible reactive protocol with low overhead message complexity which finds an optimal cost-efficient path from $s$ to $t$ with respect to cost $g$.

### 2.1.3 Summary of Results

We first present the basic ad hoc-VCG protocol for routing in ad hoc networks with selfish devices, which is incentive-compatible and reactive, and finds an optimal cost-efficient path. We, then, introduce the improved ad hoc-VCG protocol that shares the properties of the basic ad hoc-VCG protocol, and additionally exhibits better performance with respect to the number of sent overhead messages under a stronger assumption. Finally, we evaluate the performance of the two protocols by means of simulation.

Section 2.2 describes the basic ad hoc-VCG protocol and the improved ad hoc-VCG protocol. The analysis of the protocols with respect to different requirements can be found in Section 2.3. In Section 2.4, we describe the experiments and their results. Most parts of this chapter have been published in [3].

### 2.2 The Ad Hoc-VCG Protocols

This section proposes the basic ad hoc-VCG and the improved ad hoc-VCG protocol. Both protocols are a variation of the classic VCG mechanism (see Section 1.3 for a brief description of the classic VCG mechanism).
2.2. The Ad Hoc-VCG Protocols

2.2.1 Basic Ad Hoc-VCG Protocol

The basic ad hoc-VCG protocol first computes a path between the source device $s$ and the destination device $t$, and then routes the data packets from $s$ to $t$ along this path. Basic ad hoc-VCG consists of the following two phases:

- *Route Discovery*
  
  Initiated by the source device $s$, edge costs are computed and collected by the destination device $t$ in a flooding scheme. The destination computes a shortest path as well as the payments to be made to the intermediate devices for each forwarded data packet, and sends this information back to the source.

- *Data Transmission*
  
  The source device $s$ sends data packets to the destination device $t$ along the computed path. The destination receives the data packets and acknowledges the received data packets.

We next present these two phases in more detail.

**Route discovery**

The route discovery phase determines a path from the source device $s$ to the destination device $t$, and it mainly follows [13, 29]. Whenever $s$ wants to communicate with $t$, it initiates the route discovery process by broadcasting a ROUTE REQUEST packet. This packet contains the following items which are cryptographically signed in order to prevent other devices from altering them:

1. A sequence number $scq_{s,t}$
2. The identification $id_s$ of the source device $s$
3. The identification $id_t$ of the destination device $t$
4. The emission power $P_{s}^{emit}$ of the emitted signal
5. The cost-of-energy value $e_s$

Every device except the source $s$ stores a local view of the ad hoc graph. The local view is empty before the first route request packet arrives. Whenever a device $v$ other than $s$ or $t$ receives the ROUTE REQUEST packet from a device $u$, it executes the following algorithm:

- Determine the power $P_{u,v}^{\text{rec}}$ at which the packet was received.

- Compute the minimum power required for device $u$ to transmit to device $v$ as $P_{u,v}^{\text{min}} = \frac{P_u^{\text{emit}}}{P_{u,v}^{\text{rec}}} \cdot P_{u,v}^{\text{min}}$, where $P_u^{\text{emit}}$ is taken from the received ROUTE REQUEST packet.

- Compute the edge cost for $(u, v)$ as $g(u, v) = e_u \cdot P_{u,v}^{\text{min}}$ where $P_{u,v}^{\text{min}}$ is taken from the previous step and $e_u$ is taken from the received ROUTE REQUEST packet.

- Check whether the received ROUTE REQUEST packet contains information about an edge, including $(u, v)$, not yet contained in the local view of the ad hoc graph. If such information is found, then add new edges to the local view of the ad hoc graph. Otherwise, drop the packet and stop the algorithm.

- Replace the emission power value $P_u^{\text{emit}}$ in the received ROUTE REQUEST packet by $P_{u,v}^{\text{min}}$, append to the packet the identification $id_u$, the emission power $P_v^{\text{emit}}$ which the device will later use to broadcast, and the cost-of-energy value $e_v$, signed by $v$.

- Rebroadcast the created ROUTE REQUEST packet with power $P_v^{\text{emit}}$.

The ROUTE REQUEST packet contains the fields as indicated in Figures 2.2.1.a and 2.2.1.b.

The destination device $t$ collects all arriving packets, and constructs its local view of the ad hoc graph. Once it has collected all information it

\[\text{\footnotesize\textsuperscript{6}Alternatively, we could only include edges of the path contained in the received packet that are actually new information and neglect other edges.}\]
computes a shortest $s-t$-path $SP(s, t) = \{s = \sigma(1), \sigma(2), \ldots, \sigma(l(\sigma)) = t\}$ in its local view of the ad hoc graph. If there is more than one shortest path then the destination randomly chooses one. Recall that $|SP(s, t)|$ denotes the total cost of the shortest path $SP(s, t)$. In order to compute the VCG payments that are to be made to the intermediate devices, the destination also computes for each device $\sigma(i)$, $1 < i < l(\sigma)$, a shortest replacement path $SP^{-\sigma(i)}(s, t)$ in its local view of the ad hoc graph.\footnote{By our $2$-connectedness assumption, these replacement paths always exist.} We then define the VCG payment $pay_{\sigma(i)}$ for an intermediate device $\sigma(i)$ to be:

$$pay_{\sigma(i)} = |SP^{-\sigma(i)}(s, t)| - |SP(s, t)| + e_{\sigma(i)} \cdot P_{\sigma(i), \sigma(i+1)}^{\text{emit}}.$$ 

In words, $pay_{\sigma(i)}$ is the difference between the total cost of the shortest $s-t$-path when device $\sigma(i)$ does not exist, and the total cost of the shortest $s-t$-path without the cost incurred by $\sigma(i)$. All devices not on the chosen path $SP(s, t)$ receive no payment.

To illustrate how the payments are computed, consider the example graph with six devices in Figure 2.2.2. The number on the edges indicate
the computed cost to transmit a unit-size data packet along this edge. The shortest path from source $s$ to destination $t$ is $SP(s, t) = \{s, v_2, v_3, t\}$ with cost 10. The shortest replacement path for device $v_2$ is $SP^{-v_2}(s, t) = \{s, v_1, v_4, t\}$ with cost 14, and the shortest replacement path for device $v_3$ is $SP^{-v_3}(s, t) = \{s, v_2, v_4, t\}$ with cost 12. Thus, the devices receive the following VCG payments:

\[
\begin{align*}
\text{pay}_{v_1} &= 0, & \text{pay}_{v_2} &= 14 - 10 + 2 = 6, \\
\text{pay}_{v_3} &= 12 - 10 + 3 = 5, & \text{pay}_{v_4} &= 0.
\end{align*}
\]

Figure 2.2.2: An example ad hoc graph.

After computing the shortest path and the payments, destination device $t$ creates a ROUTE REPLY packet containing the identifications of the devices on the shortest path $SP(s, t)$, the corresponding minimum required emission powers $P^\text{min}_{\sigma(i),\sigma(i+1)}$, and the VCG payments $\text{pay}_{\sigma(i)}$ (see Figure 2.2.1.c). The ROUTE REPLY packet is sent back to source device $s$ along the reversed order of the discovered route. In order to prevent intermediate devices from altering the information in the ROUTE REPLY packet to their advantage, the destination signs this packet with a digital signature, which ensures that the source receives the correct data.
2.2. The Ad Hoc-VCG Protocols

As an aside, note that, unlike [13, 29], an intermediate device rebroadcasts the ROUTE REQUEST packet whenever the path in the packet contains an edge not known before to the intermediate device. This extra information is needed to compute the replacement paths. The protocol in [13, 29] only rebroadcasts the ROUTE REQUEST packet when the path in the packet forms a shorter path than known before to the intermediate device.

Data Transmission

In the data transmission phase, the data packets are sent along the shortest path \( SP(s, t) \). All devices use the minimum emission power specified in the ROUTE REPLY packet to forward the packages to the next device on \( SP(s, t) \).

In the example of Figure 2.2.2, the source device sends the data packets to device \( v_2 \) causing transmission cost of 5. In addition, the source also needs to pay amounts \( pay_{v_2} \) and \( pay_{v_3} \) per unit-size data packet to devices \( v_2 \) and \( v_3 \), respectively, for their forwarding service. Thus, the overall cost for \( s \) amounts to \( 5 + 6 + 5 = 16 \).

2.2.2 Improved Ad Hoc-VCG Protocol

The improved variant of the basic ad hoc-VCG protocol is also a reactive protocol consisting of the two phases route discovery and data transmission. Its key feature lies in a comparatively small overhead message complexity. This is achieved by borrowing techniques from geometric routing in ad hoc networks. To this end, we need two further assumptions. First, we assume that the minimum Euclidean distance between any two devices as well as the minimum cost-of-energy value of any device is at least one. This implies that all edges in \( E \) have cost at least one, that is, \( g(u, v) \geq 1 \) for all \( (u, v) \in E \). This minimum edge cost assumption is a generalization of the \( \Omega(1) \)-model (see [52] for details). Second, we assume that there exists a global time that is known to every device. In fact, the route discovery phase is the only phase diverging from the basic ad hoc-VCG. Hence, we
only describe this phase. The data transmission phase works as in the basic ad hoc-VCG protocol.

Route Discovery

The route discovery phase tries to avoid that a route request floods the whole network although the destination is close to the source. To this end, the protocol runs in rounds. In the first round, the route request packet is only broadcast to devices with shortest path cost of at most one. If the route request does not reach the destination, then the second round starts. In this round, the route request packet reaches devices with shortest path cost of at most two. In this way, the protocol increases the path cost threshold in every round until the destination is found. A single round works similarly as the whole protocol in the basic ad hoc-VCG protocol except that some additional steps are included to prevent a device from reusing information from an earlier round.

More precisely, the protocol works as follows. When the source device $s$ has a communication request, it broadcasts a ROUTE REQUEST packet that contains the following items, which are cryptographically signed in order to prevent other devices from altering them:

1. An expiration time $\Upsilon_{s,t}$
2. The identification $id_s$ of the source device $s$
3. The identification $id_t$ of the destination device $t$
4. A path cost threshold $\omega$
5. The emission power $P_{s}^{emit}$ of the emitted signal
6. The cost-of-energy value $c_s$

The first four items are signed as a single item, thus preventing other devices from falsely combining this information. The path cost threshold $\omega$ is set to one.
2.2. The Ad Hoc-VCG Protocols

Intermediate devices build up a local view of the ad hoc graph. More precisely, an intermediate device $v$ that receives a ROUTE REQUEST packet from another device $u$ executes the following algorithm:

- Check the expiration time in the received ROUTE REQUEST packet and drop the packet if the time has expired.

- Determine the power $P_{u,v}^{rec}$ at which the packet was received.

- Compute the minimum power required for device $u$ to transmit to device $v$ as $P_{u,v}^{min} = \frac{P_u^{emit}}{P_{u,v}^{rec}} \cdot P_{u,v}^{rec}$, where $P_u^{emit}$ is taken from the received ROUTE REQUEST packet.

- Compute the edge cost for $(u, v)$ as $g(u, v) = e_u \cdot P_{u,v}^{min}$ where $P_{u,v}^{min}$ is taken from the previous step and $e_u$ is taken from the received ROUTE REQUEST packet.

- Check whether the received ROUTE REQUEST packet contains information about an edge, including $(u, v)$, not yet contained in the local view of the ad hoc graph, or contained in the local view with a lower expiration time. If such information is found, then add new edges to the local view of the ad hoc graph marking them as "not yet forwarded". Otherwise, drop the packet and stop the algorithm.

- For each edge $(v_i, v_j)$ marked as "not yet forwarded" in the local view of the ad hoc graph, compute the shortest path from source $s$ to device $v_i$ and the shortest path from $v_j$ to device $v$ in the local view. If the cost of the path from source $s$ to $v$ via edge $(v_i, v_j)$ is less than the path cost threshold $\omega$, mark edge $(v_i, v_j)$ as "to be forwarded".

- Create a ROUTE REQUEST packet containing the following items:
  
  1. The expiration time $\Upsilon_{s,t}$, the identification $id_s$ of source device $s$, the identification $id_t$ of destination device $t$, and the path cost threshold $\omega$ from the received packet, all signed by $s$
  
  2. The own identification number $id_v$, signed by $v$
3. The emission power $P_{v}^{emit}$ which the device will later use to broadcast, signed by $v$

4. The cost-of-energy value $e_v$, signed by $v$

5. The edge cost information, that is, the identifications $id_{v_i}$ and $id_{v_j}$, the emission power $P_{v_i}^{emit}$, the receiving power $P_{v_i,v_j}^{rec}$, and the cost-of-energy value $e_{v_i}$ for each edge $(v_i, v_j)$, marked as "to be forwarded", signed by $v$

The fields that are signed by $v$ are all signed together with expiration time $\Upsilon_{s,t}$ to indicate that these values are valid for the current round.

- Rebroadcast the created ROUTE REQUEST packet with power $P_{v}^{emit}$.

The algorithm above takes care that intermediate devices forward edge cost information only if the edge lies on a path that has a chance of being short enough to reach the destination without exceeding the threshold $\omega$.

The destination device $t$ performs similar steps as in the basic ad hoc VCG protocol. It waits for ROUTE REQUEST packets, and update its local view each time a packet arrives. Once the expiration time in the ROUTE REQUEST packet has expired, the destination computes a shortest $s-t$-path $SP(s, t) = \{s = \sigma(1), \sigma(2), \ldots, \sigma(l(\sigma)) = t\}$, and a shortest replacement path $SP^{\sigma(i)}(s, t)$ for each intermediate device $\sigma(i)$, $1 < i < l(\sigma)$, on the shortest path. If any replacement path has infinite cost, then $t$ stops, clears its local view of the ad hoc graph, and waits for ROUTE REQUEST packets with a later expiration time. Otherwise, the destination calculates the payment $pay_{\sigma(i)}$ that is to be made to intermediate device $\sigma(i)$ on the shortest path $SP(s, t)$ for each data packet according to 2.2.1. Next, it creates a signed ROUTE REPLY packet containing the identifications of the devices on the shortest path, the minimum required emission power, and the payments, and sends the ROUTE REPLY back to the source device $s$.

Source device $s$ waits a predetermined amount of time after the expiration time $\Upsilon_{s,t}$ has passed. If it does not receive a ROUTE REPLY packet from the destination during this time, it starts the next round of the route discovery phase. To this end, it doubles the path cost threshold $\omega$, sets a
new expiration time $\Upsilon_{s,t}$ and broadcasts a new ROUTE REQUEST packet. If $s$ does receive a ROUTE REPLY packet, it knows it can begin sending data packets.

2.3 Analysis of the Ad Hoc-VCG Protocols

This section analyzes the basic and improved ad hoc-VCG protocol with respect to the three properties of incentive-compatibility, efficiency, and overhead message complexity. The reactive nature of the protocols is given by construction. In addition, we investigate the issue of overcharging, that is, the cost for the source including the payments in comparison to the actual cost of the selected path.

2.3.1 Incentive-Compatibility

Incentive-compatibility holds for the basic ad hoc-VCG protocol if each device maximizes its utility by following the protocol specification described in Section 2.2.1, assuming that all other devices follow the protocol specification. Following the protocol specification means for basic ad hoc-VCG that a device $v$

1. Declares its true cost-of-energy value $e_v$ and its true emission power $P_{v}^{em}$,

2. Correctly computes and declares the minimum emission power $P_{u,v}^{\text{min}}$ of all its neighbors $u$, and

3. Correctly rebroadcasts the information about the cost of all edges contained in the received ROUTE REQUEST packets received without alterations or intentional dropping.

In a game-theoretic sense, this means that the strategies satisfying the three items listed above forms a Nash equilibrium. We refer to these three items as “cheating possibilities”.
For the analysis we distinguish between the source device $s$, the destination device $t$ and other devices. By assumption, the source $s$ declares its true cost-of-energy value $e_s$ and its true emission power $P^{emit}_s$. Without this assumption, the source device might have reason to underdeclare.\footnote{As an example for such a situation, assume that the source device knows that it will end up paying a large amount due to the true graph topology, which exceeds even the true cost that it would incur by communicating with the destination in a single hop. If the source then underdeclares its cost-of-energy value, it can turn the single hop connection to the destination into the shortest path, thus end up paying less.} The other two cheating possibilities cannot be exploited by $s$. Thus, the source will follow the protocol.

The destination device $t$ has no cost, and its aim is to receive the data packets in our model. Hence, it will follow the protocol as well. For all other devices, subsequently called devices, we consider each cheating possibility in a lemma, and then argue that combining cheating possibilities is also not advantageous.

**Lemma 2.1.** If all other devices follow the protocol, then it is best for a device $u$ to declare its true cost-of-energy value $e_u$ and its true emission power $P^{emit}_u$.

**Proof.** Recall that the minimum required emission power of $u$ is computed according to $P^{\min}_{u,v} = \frac{P^{emit}_u P^{\min}_{u,v}}{P^{prec}_{u,v}}$, and that the cost of an edge $(u, v)$ is evaluated to $e_u \cdot P^{\min}_{u,v}$. As all other devices follow the protocol, the declarations of $P^{emit}_u$ and $e_u$ directly determine the cost $g(u, v)$. Moreover, note that a device cannot declare $e_u$ or $P^{emit}_u$ for each single outgoing edge, and consequently changes the cost of all its outgoing edges by misdeclaring either of the two values. Hence, this cheating possibility corresponds to a classical misdeclaration in mechanism design (see for example [57]).

Suppose that $(u, v)$ is part of the shortest path when declaring true values. Then, the utility of $u$ is positive. If $u$ overdeclares $g(u, v)$ by some combination of declaring $P^{emit}_u$ and $e_u$, then either $(u, v)$ drops out of the shortest path or it remains part of it, but the utility of $u$ remains the same because of the VCG nature of the payments. If $u$ underdeclares $g(u, v)$
by some combination of declaring $P_u^{\text{emit}}$ and $e_u$, then $(u, v)$ is still on the shortest path, but its utility also remains the same.

Suppose that $(u, v)$ is not part of the shortest path when declaring true values. Then, the utility of $u$ is zero. If $u$ overdeclares $g(u, v)$, then $(u, v)$ does not become part of the shortest path, and the utility of $u$ remains zero. If $u$ underdeclares $g(u, v)$, then either $(u, v)$ is still not on the shortest path, or it becomes part of the shortest path. In the former case, the utility of $u$ remains zero. In the latter case, the utility of $u$ is negative, as its cost is larger than its received payment.

\[ \square \]

**Lemma 2.2.** If all other devices follow the protocol, then it is best for a device $v$ to declare the true minimum emission power $P_{u,v}^{\text{min}}$ of all its neighbors $u$.

**Proof.** As all other devices follow the protocol, the declaration of $P_{u,v}^{\text{min}}$ directly determines the cost $g(u, v)$, that is, the transmission cost along $(u, v)$ of device $u$. We distinguish the two cases of overdeclaring and underdeclaring.

We start with overdeclaring. Suppose that $(u, v)$ is part of the shortest path when $v$ declares the true value. Then, the utility of $v$ is positive. Overdeclaring $P_{u,v}^{\text{min}}$ induces three possibilities. If the shortest path remains the same, then the total cost of the shortest path increases, yielding a lower utility for $v$. If the shortest path changes, then device $v$ either is not on the shortest path anymore, or it still is. In the former case, the utility for $v$ is zero. In the latter case, the edge $(u, v)$ is not part of the shortest path anymore, and the total cost of the new induced shortest path are larger than before yielding a lower utility for $v$. If $(u, v)$ is not part of the shortest path when $v$ declares the true value, then overdeclaring has no effect on $v$’s utility.

By underdeclaring $P_{u,v}^{\text{min}}$, the device $v$ may either enter the shortest path if $v$ is not on the shortest path under true declaration, or it may decrease the cost of the shortest path if $v$ is already on the shortest path under true declaration. Thus, the utility of $v$ has the potential to increase in both
cases. However, if $u$ emits the data packets with the underdeclared emission power, device $v$ is outside the transmission range of $u$. Thus, the data packets do not arrive at the destination $t$, and no payment is delivered to $v$. 

We also have to guarantee that no device can take advantage of the distributed nature of the protocol [60, 33]. The declarations are not sent directly to the destination device (which computes a shortest path and the payments) but forwarded by other devices. These devices can potentially manipulate the declarations of the preceding devices on the path of the ROUTE REQUEST packet.

**Lemma 2.3.** If all other devices follow the protocol, then it is best for a device $v$ to correctly rebroadcast the information about the cost of all edges contained in the received ROUTE REQUEST packets without alterations or intentional dropping.

**Proof.** Suppose device $v$ decreases the edge cost $g(v_i, v_j)$ contained in a ROUTE REQUEST packet. We distinguish the following cases:

- If edge $(v_i, v_j)$ is on the shortest path before the cost decrease, then it will also be on it after the alteration. However, the transmission of the data packets will fail as $v_j$ is outside the transmission range of $v_i$, yielding no payment for $v$.

- If device $v$ is on the shortest path $SP(s, t)$ and edge $(v_i, v_j)$ is on the shortest replacement path $SP^{-v}(s, t)$ for device $v$ before the decrease, then the cost decrease will result in a smaller payment for device $v$, thus reducing its utility.

- If device $v$ is on the shortest path $SP(s, t)$ and edge $(v_i, v_j)$ is not on $SP(s, t)$ nor on $SP^{-v}(s, t)$, then the decrease will either have no effect, or decrease the cost of $SP^{-v}(s, t)$, or knock device $v$ off the shortest path, thus reducing its utility.

- If device $v$ is not on $SP(s, t)$, reducing the edge cost may put $v$ onto the shortest path. However, the transmission of the data packets will
2.3. Analysis of the Ad Hoc-VCG Protocols

fail as \( v_j \) is outside the transmission range of \( v_i \), thus no payment is delivered to \( v \).

Thus, device \( v \) cannot increase its utility by decreasing the cost of an edge. Now, suppose device \( v \) increases the edge cost \( g(v_i, v_j) \). This action might increase \( v \)'s utility if either:

- device \( v \) lies on \( SP(s, t) \) and edge \((v_i, v_j)\) lies on \( SP^{-v}(s, t) \) before the increase, or

- device \( v \) does not lie on \( SP(s, t) \) and edge \((v_i, v_j)\) lies on \( SP(s, t) \) before the increase.

However, in the first case, all devices on \( SP^{-v}(s, t) \) will forward their ROUTE REQUEST packets containing the true edge cost \( g(v_i, v_j) \), and the cheating device \( v \) is not on \( SP^{-v}(s, t) \), thus enabling the destination device to simply ignore the increased edge cost. Similarly, in the second case, the devices on \( SP(s, t) \) will forward their ROUTE REQUEST packets containing edge cost \( g(v_i, v_j) \), and the cheating device \( v \) is not on \( SP(s, t) \).

Finally, if device \( v \) intentionally drops ROUTE REQUEST packets, all the information in these packets, except for the edge costs of incoming and outgoing edges from \( v \), will find its way to the destination through the path \( SP^{-v}(s, t) \).

\[ \square \]

In order to see that combining the cheating possibilities also does not increase the utility of device \( v \), first assume that \( v \) is not on the shortest path \( SP(s, t) \). Then, device \( v \) has two possibilities to move itself onto the shortest path. It can either try to increase the cost of \( SP(s, t) \), or it can try to decrease the cost of a path containing \( v \). The former fails as the devices on \( SP(s, t) \) report truthfully. The latter results in a communication failure as a device on the new shortest path is outside the transmission range of its predecessor, or it results in a negative utility for \( v \) if it underdecls its own cost. Now, assume that \( v \) is on the shortest path \( SP \). Then, \( v \) again has two possibilities to increase its utility. It can either try to increase the cost of the path \( SP^{-v}(s, t) \), which fails as the devices on \( SP^{-v}(s, t) \) report truthfully, or it can try to underdeclare the cost of the other devices on
$SP(s, t)$ while overdeclaring its own cost, which will – once again – result in a communication failure as a device will be outside the transmission range of its predecessor.

As a last part of the route discovery phase, the intermediate devices on the shortest path need to forward a ROUTE REPLY packet to the source along the reverse shortest path. Since the ROUTE REPLY packet is signed, the devices cannot change the content of the packet and simply dropping it does not increase their utility as this will prevent the communication from taking place. Thus, the intermediate devices correctly execute this step as well.

During the data transmission phase, intermediate devices actually forward data packets as their account will only be credited for each data packet received by the destination. Moreover, the source and the destination device are interested in the communication by assumption. Hence, the data transmission phase is incentive-compatible as well.

Thus, we have shown the following theorem:

**Theorem 2.4.** Basic ad hoc-VCG is incentive-compatible.

The improved ad hoc-VCG protocol has the same “cheating possibilities” as the basic ad hoc-VCG protocol plus the additional possibility to alter either the expiration time or the path cost threshold. As both the expiration time and the path cost threshold are cryptographically protected against alteration, the incentive-compatibility of improved ad hoc-VCG follows from the incentive-compatibility of basic ad hoc-VCG.

### 2.3.2 Efficiency

Cost-efficiency of both ad hoc-VCG protocol variants follows immediately from the specification of ad hoc-VCG, if incentive-compatibility is guaranteed. The destination device in ad hoc-VCG collects all edge costs of the ad hoc graph, and computes the most cost-efficient routing path based on these edge costs.
2.3. Analysis of the Ad Hoc-VCG Protocols

Corollary 2.5. Basic ad hoc-VCG and improved ad hoc-VCG route along the most cost-efficient path between a source device \( s \) and a destination device \( t \).

If we additionally assume that all devices have the same cost-of-energy value (i.e., \( c_v \) is constant for all devices \( v \in V \)), then the mechanism also chooses the most energy-efficient path.

2.3.3 Overhead Message Complexity

As for the overhead of the route discovery phase in the basic ad hoc-VCG protocol, all devices need to store their local view of the ad hoc graph in order to determine whether an incoming ROUTE REQUEST packet contains new information that needs to be forwarded. Thus, we need a data structure similar to the energy aware link cache introduced in [29]. Each of the \( n \) devices (except \( s \) and \( t \)) may need to forward \( O(n^2) \) ROUTE REQUEST packets containing at least one new edge with its cost, resulting in a total of \( O(n^3) \) control packets sent in this phase. In order for the basic ad hoc-VCG protocol to be incentive-compatible, we must guarantee to compute the cost of all edges. This partly explains the rather large overhead.

Next, we derive an upper bound with respect to overhead message complexity for the improved ad hoc-VCG protocol.

Lemma 2.6. The overhead message complexity of the improved ad hoc-VCG protocol is \( O(|SP_{max}^{-\sigma}(s,t)|^6) \) where \( SP_{max}^{-\sigma}(s,t) \) is the shortest replacement path with largest cost, taken over all devices on the shortest path \( SP(s,t) \).

Proof. Recall the assumption that the minimum cost for any edge is one, that the minimum cost-of-energy value is normed to one, and that the devices are placed in the plane such that their minimum distance is at least one. Consider a disk (the boundary and its interior) with radius \( 1 \). Since the minimum distance between each pair of devices is at least one, there are at most 7 devices inside such a disk. To upper bound the number of devices inside a disk with radius \( \omega \), we derive an upper bound on the number
of disks with radius one needed to cover a disk with radius $\omega$. Consider the smallest square enclosing the disk with radius $\omega$. Inside this square, we place disks with radius one on a grid such that the center of a disk lies on each grid point, and the distance between two adjacent grid points is equal to one. This placement guarantees that the whole disk with radius $\omega$ is covered, and the number of disks is equal to $(2\omega + 1)^2$. Thus, the number of devices inside the disk with radius $\omega$ is bounded by $7(2\omega + 1)^2$ respectively $O(\omega^2)$.

Further, there can be $O(\omega^4)$ edges between these $O(\omega^2)$ devices. Since the edge cost between any two devices is at least one, there is no path from the source to any device outside this disk with cost less than $\omega$, because a distance greater than $\omega$ must be covered. Consequently, at most $\omega^2$ devices will be involved in the route discovery for a path cost threshold $\omega$. Each of these devices forwards $O(\omega^4)$ different edge costs resulting in a total number of $O(\omega^6)$ control packets for a fixed path cost threshold $\omega$.

The path cost threshold is initially set to one and then doubled in each round. Let $\omega_\ell = 2^{\ell-1}$ denote the value of $\omega$ in the $\ell$-th round. Further, let $SP_{\max}^{-\sigma}(s, t)$ denote the shortest replacement path $SP^{-\sigma(i)}(s, t)$ with largest cost for a shortest path $SP(s, t) = \{s, \sigma(1), \ldots, \sigma(|\sigma|-1), t\}$, and $\bar{\ell}$ be the smallest integer $\ell$ such that $\omega_{\bar{\ell}} \geq |SP_{\max}^{-\sigma}(s, t)|$. Thus, there will be $\bar{\ell}$ rounds in route discovery. In the $\ell$-th round, $O(\omega_\ell^6)$ control packets will be sent. Summing up over all rounds, we get

$$\sum_{\ell=1}^{\bar{\ell}} \omega_\ell^6 = \sum_{\ell=1}^{\bar{\ell}} (2^{\ell-1})^6 = \frac{(2^6)^{\bar{\ell}} - 1}{2^6 - 1} \in O((2^6)^{\bar{\ell}}) = O(|SP_{\max}^{-\sigma}(s, t)|^6)$$

overhead messages sent around in the route discovery phase.

---

\[ This \text{ bound is certainly not optimal, but enough for our purposes here. } \]
2.3. Analysis of the Ad Hoc-VCG Protocols

The overhead message complexity is independent of the total number of devices in the network, which is unlike the basic ad hoc-VCG protocol.

If we have a maximum emission power \( P_{\text{emit}}^{\max} \) with which the devices can emit, we can express the overhead message complexity in an alternative way. Assume that the radius \( r_{\text{max}} \) (resulting from \( P_{\text{emit}}^{\max} \)) is the maximum distance between two devices with a direct link. Then, a device can have at most \( r_{\text{max}}^2 \) neighbors (using a similar argument as before). Thus, the \( \omega^2 \) devices in the disk of radius \( \omega \) around a device will have to report \( O(\omega^2 r_{\text{max}}^2) \) edges, resulting in an overall overhead message complexity of \( O(\left|SP_{\text{max}}^{-\sigma}(s, t)\right|^{\omega^2} r_{\text{max}}^2) \). If \( r_{\text{max}} \) is a small constant, which may be the case in some scenarios, the overhead message complexity is clearly reduced.

2.3.4 Overcharging

Both ad hoc-VCG protocol variants force the source device \( s \) to pay all intermediate devices on the shortest path \( SP(s, t) = \{ \sigma(1), \ldots, \sigma(l(\sigma)) \} \). To guarantee incentive-compatibility in the VCG protocols, the payment to each device is at least the true cost of the device. This raises the question about how much larger the sum of these payments can be in comparison to the actual costs of all devices. To this end, we consider the overcharge ratio \( \frac{C_{VCG}}{|SP(s, t)|} \), where \( C_{VCG} \) is the total cost for the source, and \( |SP(s, t)| \) is as usual the total cost of the shortest path. Next, we prove an upper bound on the overcharge ratio. The total cost \( C_{VCG} \) for the source consists of the payments for all devices on the shortest path plus the cost to transmit the data packets to the first device on the shortest path.

\[
C_{VCG} = e_s \cdot P_{s, \sigma(2)}^{\min} + \sum_{i=2}^{l(\sigma)-1} p_{\sigma \gamma \sigma(i)}
\]

\[
= \sum_{i=2}^{l(\sigma)-1} |SP^{-\sigma(i)}(s, t)| - (l(\sigma) - 3)|SP(s, t)|.
\]
The actual total cost \(|SP(s, t)|\) faced by the devices on the shortest path is
\[
|SP(s, t)| = \sum_{i=1}^{l(\sigma)-1} e_{\sigma(i)} \cdot P_{\sigma(i), \sigma(i+1)}^{\min}.
\]

**Lemma 2.7.** Let \(e_{\max}\) be the maximum, and \(e_{\min}\) be the minimum cost-of-energy value of any device on the shortest path \(SP(s, t)\), and let \(\alpha\) be the path loss exponent with which signal strength is lost. Then
\[
\frac{C_{VGCG}}{|SP(s, t)|} \leq 2^{\alpha+1} \frac{e_{\max}}{e_{\min}}.
\]

**Proof.** The key idea of the proof is to bound the cost of the shortest replacement paths \(SP^{-\sigma(i)}(s, t)\) by analyzing the path
\[
\{s, \sigma(2), \ldots, \sigma(i - 1), \sigma(i + 1), \ldots, \sigma(l(\sigma) - 1), t\},
\]
which is equivalent to the shortest path \(SP(s, t)\), but overhops device \(\sigma(i)\) by linking \(\sigma(i - 1)\) directly to \(\sigma(i + 1)\).\(^{10}\) Thus, we have
\[
|SP^{-\sigma(i)}(s, t)| \leq |SP(s, t)| + e_{\sigma(i-1)} \cdot P_{\sigma(i-1), \sigma(i+1)}^{\min} - (e_{\sigma(i-1)} \cdot P_{\sigma(i-1), \sigma(i)}^{\min} + e_{\sigma(i)} \cdot P_{\sigma(i), \sigma(i+1)}^{\min}),
\]
for \(1 < i < l(\sigma)\). In a next step, we derive an upper bound for \(P_{\sigma(i-1), \sigma(i+1)}^{\min}\):

\[
P_{\sigma(i-1), \sigma(i+1)}^{\min} = \frac{P_{\sigma(i-1), \sigma(i+1)}^{\min}}{\Gamma} \cdot |\sigma(i - 1), \sigma(i + 1)|^{\alpha}
\]
\[
\leq \frac{P_{\sigma(i-1), \sigma(i+1)}^{\min}}{\Gamma} \cdot (|\sigma(i - 1), \sigma(i)| + |\sigma(i), \sigma(i + 1)|)^{\alpha}
\]
\[
\leq \frac{P_{\sigma(i-1), \sigma(i+1)}^{\min}}{\Gamma} \cdot 2^{\alpha} (|\sigma(i - 1), \sigma(i)|^{\alpha} + |\sigma(i), \sigma(i + 1)|^{\alpha})
\]
\[
= 2^{\alpha} (P_{\sigma(i-1), \sigma(i)}^{\min} + P_{\sigma(i), \sigma(i+1)}^{\min})
\]

\(^{10}\)Unfortunately, a mechanism that always pays according to this rule is not truthful. However, for our analytical purpose we can use this path. This requires that this edge actually exists, which is the case if the maximal emission power level of each device is infinite, but is not necessarily the case for any 2-connected graph as introduced in Section 2.1.2.
Using (2.3.2) in (2.3.1) we get:

\[
|SP^{-\sigma(i)}(s, t)| \leq |SP(s, t)| + 2^\alpha \cdot e_{\sigma(i-1)} \left( P_{\sigma(i-1), \sigma(i)}^{\text{min}} + P_{\sigma(i), \sigma(i+1)}^{\text{min}} \right) \\
- (e_{\sigma(i-1)} \cdot P_{\sigma(i-1), \sigma(i)}^{\text{min}} + e_{\sigma(i)} \cdot P_{\sigma(i), \sigma(i+1)}^{\text{min}})
\]

For the total cost \( C_{VCG} \), we obtain

\[
C_{VCG} \leq \sum_{i=2}^{l(\sigma)-1} \left( |SP(s, t)| + 2^\alpha \cdot e_{\sigma(i-1)} \left( P_{\sigma(i-1), \sigma(i)}^{\text{min}} + P_{\sigma(i), \sigma(i+1)}^{\text{min}} \right) \\
- (l(\sigma) - 3)|SP| \right)
\]

\[
= |SP| + \sum_{i=2}^{l(\sigma)-1} 2^\alpha \cdot e_{\sigma(i-1)} \left( P_{\sigma(i-1), \sigma(i)}^{\text{min}} + P_{\sigma(i), \sigma(i+1)}^{\text{min}} \right) \\
- \sum_{i=2}^{l(\sigma)-1} \left( e_{\sigma(i-1)} \cdot P_{\sigma(i-1), \sigma(i)}^{\text{min}} + e_{\sigma(i)} \cdot P_{\sigma(i), \sigma(i+1)}^{\text{min}} \right)
\]

\[
\leq \sum_{i=2}^{l(\sigma)-1} 2^\alpha \cdot e_{\sigma(i-1)} \left( P_{\sigma(i-1), \sigma(i)}^{\text{min}} + P_{\sigma(i), \sigma(i+1)}^{\text{min}} \right).
\]

For the overcharge ratio, we thus have:

\[
\frac{C_{VCG}}{|SP(s, t)|} \leq 2^\alpha \frac{e_{\text{max}}}{e_{\text{min}}} \sum_{i=2}^{l(\sigma)-1} P_{\sigma(i-1), \sigma(i)}^{\text{min}} + \sum_{i=1}^{l(\sigma)-1} P_{\sigma(i), \sigma(i+1)}^{\text{min}} \\
\leq 2^\alpha + 1 \frac{e_{\text{max}}}{e_{\text{min}}}.
\]

The path loss exponent \( \alpha \) is usually assumed to be at most six. The ratio \( \frac{e_{\text{max}}}{e_{\text{min}}} \) between the maximum and minimum cost-of-energy value can
be assumed to be reasonably low (in fact, we could impose a minimum and maximum cost-of-energy value upon all devices without losing truthfulness). Hence, Lemma 2.7 gives quite a good upper bound for the overcharge ratio. We will further substantiate this claim by looking at experimental results in Section 2.4.2. The upper bound crucially depends on the special structure of the shortest replacement paths. For general ad hoc graphs without such a structure, [8] shows that there exist instances where the overcharge ratio cannot be bounded by a constant.11

2.4 Experiments

We conduct two types of experiments. In the first set of experiments, we implement the two protocols from Section 2.2.2 in GloMoSim [41] to test the performance. In the second set of experiments, we investigate the overcharge ratio.

2.4.1 Performance

The key advantage of the improved ad hoc-VCG protocol over basic ad hoc-VCG is the reduced overhead message complexity. Our goals for the experiments are to show feasibility of the improved ad hoc-VCG protocol in decent-size ad hoc networks. In addition, we compare the performance of the improved ad hoc-VCG protocol against DSR, a standard ad hoc routing protocol for a non-selfish environment to show that the additional overhead incurred by the ad hoc-VCG protocol is not prohibitively high. Finally, we compare the performance of the improved ad hoc-VCG protocol against its basic variant, which is resilient against selfishness but does not optimize overhead message complexity.

11The reason for the bounded overcharge lies in the fact that with this kind of replacement paths the agents fulfill the so-called "agents are substitutes"-property as defined in [39], respectively have frugality ratio 1 as defined in [77].
2.4. Experiments

Implementation

We implemented the improved ad hoc-VCG protocol in GloMoSim [41]. GloMoSim is a simulation environment for wireless and wired network systems. The improved ad hoc-VCG protocol is a routing protocol and thus mainly affects the network layer. However, it also has a power control component, which – at least in GloMoSim – is located on the physical layer. Our implementation of the improved ad hoc-VCG protocol covers both phases route discovery, and data transmission, and it tries to stay as close to the protocol specification given in Section 2.2.2 as possible. There are two notable exceptions to this general rule. First, we did not implement the cryptographic gadgets (digital signatures) required by the protocol specification because any standard solution would work fine for this purpose. Moreover, digital signatures would increase the size of the control packets that are sent around, but not the number of control packets. Second, we did not implement the functionality provided by the central bank because the communication with the central bank is assumed to be possible every once in a while. The implementation of the route discovery requires to add code to the physical layer of GloMoSim to determine the power of the receiving signal and to vary the emission power of a device. The algorithms for source, destination, and intermediate devices are implemented exactly according to the protocol specification. The implementation of the data transmission phase is straightforward and again follows the specification except for the payment delivery aspects. While the improved ad hoc-VCG protocol could be combined with any protocols on layers or on functions that it does not affect, we have made the following choices in our implementation, which are mostly standard for GloMoSim, and only relevant from a technical point of view. For the physical layer, the signal propagates according to the free space model of Friss with a path loss exponent $\alpha$ of 2. CSMA is our protocol of choice for the MAC layer and the functions (other than routing) on the network layer are handled by IP.

For our experiments with DSR, we used the implementation available for GloMoSim with the same protocol choices on other layers. Basic ad hoc-VCG is implemented exactly according to the description in Sec-
As far as transceiver properties are concerned, we assume omni-directional antennas with a maximum emission power of 1 dB resulting in a maximum radius of about 125 m. The remaining antenna parameters are set to the default values in GloMoSim.

Test scenarios

We characterize a scenario by the following parameters:

- The number of devices is either 9, 25, or 49 devices.
- For the device distribution, we distinguish three cases:

  **Grid placement** Devices are distributed on the vertices of a regular grid, with the grid cells chosen to be squares of 100m edge length.

  **Random square placement** Devices are placed at random inside a square. For the scenario with 9 devices, we chose a 200m x 200m square, 400m x 400m for the 25 devices, and 600m x 600m for the 49 devices.

  **Uniform grid placement** Devices are placed randomly into the cells of a uniform grid, such that exactly one device lies in each 100m x 100m cell of the grid.

- The number of parallel sessions. We use the term session for the complete process starting when the source initiates a route request, and ending when the last data packet is delivered at the destination.

For a scenario, we pick the source-destination pair(s) and sequentially execute the improved ad hoc-VCG protocol, DSR, and the basic ad hoc-VCG protocol, where each session needs to establish a route, and transfer 100 data packets. This is repeated 100 times for each scenario.

The cost-of-energy value is set to one for all devices and there is no mobility during the course of the experiments.
2.4. Experiments

Results

Our first and main result is the feasibility of the improved ad hoc-VCG protocol. The protocol runs successfully on networks with 49 devices, which is a decent size for ad hoc networks. In terms of parallel sessions, we managed to have up to 6 parallel sessions for the protocol. Usually, the shortest paths computed by the protocol consist of about five hops.

We evaluate the performance of each protocol by looking at the overhead that the protocol incurs in each scenario. Our performance measure is the number of control packets sent throughout the network. Figures 2.4.1 to 2.4.2 present plots for the different device distribution methods with the number of devices on the x-axis and the average (over the 100 runs) number of control packets on the y-axis for the improved ad hoc-VCG and DSR for a single session. For all placement methods, improved ad hoc-VCG clearly needs more overhead than DSR. Both protocols need fewer control packets when the devices are randomly placed (Figure 2.4.2) as opposed to placing them on a grid (Figure 2.4.1) or distributing them uniformly across grid cells (Figure 2.4.3). The improved ad hoc-VCG protocol seems to be particularly vulnerable in the uniform grid case, where it performs worse than on the grid. Generally, the number of control packets for improved ad hoc-VCG is in the order of the number of devices, with each device forwarding a maximum of three control packets on average. Summarizing, the additional overhead incurred by the improved ad hoc-VCG protocol to achieve resilience against selfishness is considerable when compared to DSR's overhead, but it is not prohibitively high for decent-size networks.

The histogram for the number of control packets in the random square placement for 49 devices with one session is shown in Figure 2.4.4, with intervals for the number of control packets on the x-axis, each data point representing an interval of length 10, and the frequency (in percent) on the y-axis. This figure shows that the average value for the number of control packets for the improved ad hoc-VCG protocol is affected by a few outlying runs with more than 400 control packets exchanged. DSR does not suffer from such a problem. In fact, if we ignore these few outlying data points, the curves for DSR and improved ad hoc-VCG look quite sim-
Figure 2.4.1: Results for grid placement where VCG refers to improved ad hoc-VCG.

Figure 2.4.2: Results for random square placement where VCG refers to improved ad hoc-VCG.
Figure 2.4.3: Results for uniform grid placement where VCG refers to improved ad hoc-VCG.

iliar. Both curves have a peak in the first interval between 0 and 10 control packets exchanged, which covers the cases where source and destination are very close to each other. The DSR curve then has a regular peak at about 50 control packets, the corresponding peak for the improved ad hoc-VCG protocol curve is at about 80 control packets.

For multiple parallel sessions, we show results for only one and two sessions in Figure 2.4.5. The number of sessions is on the x-axis and the number of control packets is on the y-axis, again averaged over 100 runs, for improved ad hoc-VCG, DSR and basic ad hoc-VCG on a grid of 49 devices. The performance gain of the improved ad hoc-VCG protocol over the basic ad hoc-VCG protocol is impressive. In fact, even compared to DSR, the relative gap between DSR and improved ad hoc-VCG gets smaller with an increasing number of parallel sessions. Figure 2.4.5 also shows that the overhead of the basic ad hoc-VCG protocol is prohibitively high and it is actually the reason why we are not showing results for even more parallel sessions: our simulations for basic ad hoc-VCG did not end after reasonable time. Thus, our experiments show that the improved ad
Figure 2.4.4: Histogram of number of control packets for random square placement with 49 devices where VCG refers to improved ad hoc-VCG.

hoc-VCG protocol is practical for decent-size networks, where this is certainly not the case for the basic ad hoc-VCG protocol.

2.4.2 Overcharging

In order to compute the average overcharge ratio, we have conducted experiments in the following setup: in a first set of experiments, we randomly placed 10 devices onto a rectangular grid of size 1500 x 500. We then computed the resulting ad hoc graph with edge cost \( g(u, v) = |uv|^\alpha \) between two devices \( u \) and \( v \). We ran experiments for path loss exponents \( \alpha = \{1.5, 2, 3, 4, 5, 6\} \). We set the cost-of-energy value of all devices to one. We randomly picked a source-destination pair among the devices and computed the overcharge ratio for this pair. In a second and third set of experiments, we placed 100 devices and 500 devices, all other parameters remaining the same. We ran all three sets of experiments one thousand times for each value of \( \alpha \). Table 2.1 presents overview results, giving the average overcharge ratio, the standard deviation and the maximum for each
Grid placement, 49 devices

Figure 2.4.5: Results for multiple sessions where $iVCG$ refers to improved ad hoc-VCG, and $bVCG$ to basic ad hoc-VCG.

exponent $\alpha$ and scenario. It also indicates the number of source-destination pairs for which the source would pay less by communicating directly with the destination without intermediate devices. Figure 2.4.6 shows the distribution and density functions of the overcharge ratio that we obtained from our experiments.

From Table 2.1, we see that the average overcharge ratio roughly varies between 1.16 and 1.25 for $\alpha = 1.5$, i.e., the total cost for the source are 16 to 25 per cent more than the cost of a shortest path on average. As expected, the average increases with an increasing path loss exponent $\alpha$ as replacement paths become more expensive. The average values decrease with increasing device numbers, which is to be expected as more devices imply more possibilities for (and thus cheaper) replacement paths. However, this observation only holds for $\alpha > 5$, for smaller $\alpha$ the values do not change much with increasing the number of devices. The standard deviation of the overcharge ratio ranges from 0.08 to 5.08. It increases
<table>
<thead>
<tr>
<th>Number of devices</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exponent $\alpha$</td>
<td>1.5 2 3 4 5 6</td>
</tr>
<tr>
<td>Overcharge ratio</td>
<td></td>
</tr>
<tr>
<td>average</td>
<td>1.16 1.40 1.94 2.65 4.15 5.99</td>
</tr>
<tr>
<td>std dev</td>
<td>.14 .30 .75 1.44 2.86 5.08</td>
</tr>
<tr>
<td>maximum</td>
<td>1.62 2.45 5.06 9.68 20.09 32.90</td>
</tr>
<tr>
<td>Cheaper direct communication</td>
<td>28 42 43 53 53 57</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number of devices</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exponent $\alpha$</td>
<td>1.5 2 3 4 5 6</td>
</tr>
<tr>
<td>Overcharge ratio</td>
<td></td>
</tr>
<tr>
<td>average</td>
<td>1.25 1.52 2.10 2.82 3.68 4.834</td>
</tr>
<tr>
<td>std dev</td>
<td>.10 .20 .47 .89 1.46 2.42</td>
</tr>
<tr>
<td>maximum</td>
<td>1.59 2.24 4.49 8.57 13.55 31.32</td>
</tr>
<tr>
<td>Cheaper direct communication</td>
<td>9 9 8 12 5 10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number of devices</th>
<th>500</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exponent $\alpha$</td>
<td>1.5 2 3 4 5 6</td>
</tr>
<tr>
<td>Overcharge ratio</td>
<td></td>
</tr>
<tr>
<td>average</td>
<td>1.25 1.53 2.01 2.72 3.33 4.358</td>
</tr>
<tr>
<td>std dev</td>
<td>.08 .10 .36 .70 .86 1.70</td>
</tr>
<tr>
<td>maximum</td>
<td>1.50 2.09 4.13 7.85 10.32 16.76</td>
</tr>
<tr>
<td>Cheaper direct communication</td>
<td>0 0 0 0 0 1</td>
</tr>
</tbody>
</table>

Table 2.1: Overview of experimental results.
Figure 2.4.6: Overcharge ratio distribution and density for the 100-device scenario.
Figure 2.4.7: Overcharge ratio distribution and density for the 500-device scenario.
2.5 Conclusions and Open Problems

with the path loss exponent and clearly decreases as the number of devices increases; thus having more devices seems to decrease the spread of the overcharge ratios. The maximum overcharge ratios that were paid during the 1000 experiments are clearly below the theoretical bound of $2^{\alpha+1}$ from Theorem 2.7. The maxima clearly fall with increasing number of devices, which is further evidence that having a large number of devices offers better replacement paths. The number of experiments in which the source would have had less expenses by communicating with the source directly in a single hop heavily depends on the number of devices. Once we have 500 devices, these cases become very rare.

The distribution and density functions shown in Figure 2.4.6 for the 100-device scenario and in Figure 2.4.7 for the 500-device scenario exhibit classic behavior with a clear peak at the average overcharge ratio in the density functions only for $\alpha = \{1.5, 2\}$. For $\alpha > 3$, the density curves seem quite close to each other, while the distribution curves clearly show that the path loss exponent $\alpha$ still has a large influence on the experiment for large $\alpha$.\(^{12}\)

These experimental results show that overcharge is certainly a factor that cannot be neglected. This can be considered a proof-of-concept for our model as negligible overcharge ratios would imply that good replacement paths almost always exist, thus making cheating unattractive in the first place.

2.5 Conclusions and Open Problems

Wireless ad hoc networks form a natural example of selfish behavior. The work in this chapter contains a basic protocol for routing in such networks. In the future, more work on selfish behavior in wireless ad hoc networks is expected. Different aspects such as reducing the overhead messages,

\(^{12}\)The corresponding curves are not shown for the 10-device scenario as they do not provide additional information. However, they are somewhat different as a lot more experiments achieve an overcharge ratio of 1, which is due to the fact that the direct path is very often the shortest path with only ten devices in the network.
incorporating mobility of all devices and lossy links due to obstacles, or integrating selfish behavior on different layers in the communication stack are three interesting aspects in this direction.
Chapter 3

Strategic Placement of Devices

3.1 Introduction

This chapter studies the effect of the ad hoc-VCG protocol on the structure of a wireless network. The previous chapter demonstrated that selfish behavior is evident in some wireless networks. Due to the restricted transmission range of each wireless device and the lack of a fixed infrastructure, the communication between two devices typically goes in a multi-hop fashion along intermediate devices. If the devices belong to different entities then a device is only interested in forwarding a data packet for another device when its expenses (e.g., energy cost) are covered. The approach in Chapter 2 pays money to intermediate devices to avoid selfish behavior. The amount paid to a device \(v'\) forwarding a data packet was set to \(|SP^{-v'}(s,t)| - |SP(s,t)| + c(v',v)\), where \(|SP(s,t)|\) denotes the cost of the shortest path from a source \(s\) to a destination \(t\), \(|SP^{-v'}(s,t)|\) denotes the cost of the shortest path if device \(v'\) is taken out of the network, and \(c(v',v)\) is the transmission cost at device \(v'\) to forward a data packet to
device $v$. The profit of a device is equal to its payment minus its cost for forwarding the data packet, which amounts to $|SP_{-v'}(s, t)| - |SP(s, t)|$. This principle extends to the case where a profit-maximizing agent controls a device set $V'$ and has profit $|SP_{-V'}(s, t)| - |SP(s, t)|$. Our interest in this chapter is to take the perspective of a selfish agent entering an existing network with the question where to place its devices. This directly opens the possibility to find an optimum position in the network in the selfish view of maximizing its profit.

We focus on the same problem setting as in the previous chapter. The devices are placed in the plane, and the cost $c(v, v')$ corresponds to the square of the Euclidean distance between the consecutive devices $v'$ and $v$ along a path. Further, we assume that the communication requests for the near future are known. The goal is to determine the best position for one or more additional devices such that the profit is maximum.

Our goal is to understand the underlying optimization problem for an agent which enters a wireless network and knows the rules of the game plus the current state of the network. As such, it is a first step towards the study of a network creation game in wireless networks.

### 3.1.1 Related Work

Network upgrade problems where an existing network has to be extended such that the resulting network exhibits certain properties are classical optimization problems. Several variants of these problems with different properties, different allowed extensions et cetera have been considered in the last decades. The work closest to ours is probably the thesis by Krumke [50], who proved many complexity results for problems of the following form: given a graph, a function that specifies the cost of shortening an edge, and a cost budget, determine an optimal strategy within the budget restriction such that the total weight of a minimum spanning tree is minimized.

The idea of the approach in this chapter is in a similar vein to the work on network creation games in Fabrikant et al. [32]. The main goal of [32]
3.1. Introduction

is to explain the structure of networks constructed by independent selfish agents. Their main motivation comes from the Internet where selfish agents, the autonomous systems or Internet providers, strategically choose partners to which they build a connection. In their model, the agents are vertices in a graph, in which each vertex chooses a subset of other vertices to which it will establish a connection. The cost for each vertex depends on the number of connections it establishes and the distance to all other vertices in the resulting graph. They studied the Nash equilibria of the induced game and the price of anarchy, that is the worst case ratio between the cost of a Nash equilibrium and the cost of the global optimum. Different authors [7, 6, 44, 21, 2] continued this line of study in related network creation models. Moscibroda et al. [61] studies the quality of networks resulting from selfish behavior of peers in peer-to-peer systems. Here, peers want to maintain a close link to many other peers without storing too many peers in their own memory.

The shortest path problem is one of the basic problems in graph theory. In 1959, Dijkstra [28] already presented an algorithm to find a shortest path between two vertices in a graph with running time $O(n^2)$. Since then, the running time of this algorithm has been improved to $O(n \log n + m)$ using faster data structures. For more details, we refer to Section 4.1.1 where we give a short survey of shortest path problems and its variants.

Proximity graphs as a sparse replacement graph for a denser graph is a common concept in the wireless network community. A survey on different techniques is available in [72]. Most of the techniques concentrate on networks where all devices have the same maximal transmission range. Recently, more general models where device have different maximal transmission ranges are considered by Kapoor and Li [49] and by Li et al. [54].

Beier et al. [11] investigated the shortest path problem and the shortest path problem over a restricted number of hops for the case where the vertices are points in the plane and the cost between any two vertices is equal to the squared Euclidean distance plus a constant offset. They presented an $O(n^{1+\epsilon})$ time algorithm for the shortest path problem and a $O(kn \log n)$ algorithm for the $\kappa$-hops shortest path problem. For more general cost functions and still points in the plane as vertices, Chan and Efrat [19] de-
scribed an algorithm to find the shortest path with running time $O(n^{4/3+\epsilon})$.

### 3.1.2 Model and Notation

Essentially, we use the same model as in Chapter 2. Since the perspective in this chapter is rather a theoretical computer science perspective than an engineering one, the emphasis, and consequently the notation, is slightly different. For ease of understanding, we introduce the complete notation from scratch.

The wireless devices are represented by a set $V = \{1, \ldots, n\}$, and are embedded in the plane. A placement function $p : V \rightarrow \mathbb{R}^2$ determines the $x$- and $y$-coordinate of each device. The distance measure is the Euclidean distance, that is, the distance $|uv|$ between two devices $u$ and $v$ is equal to

$$|uv| = \sqrt{(u_x - v_x)^2 + (u_y - v_y)^2},$$

where $u_x$ respectively $u_y$ denotes the $x$- and $y$-coordinate of device $u$. When no ambiguity arises, we also use $|p_1p_2|$ for the Euclidean distance between two points $p_1$ and $p_2$ in the plane. We assume that the distance between any two devices can be determined in constant time. The transmission ranges of the devices are modeled by the transmission range function $r : V \rightarrow \mathbb{R}_+$, specifying the maximal distance $r(u)$ from device $u$ at which another device can still receive a signal from $u$ via direct communication. The set $E \subseteq V \times V$ of directed edges, with $m = |E|$, represents all device pairs that can directly communicate with one another. Thus, there is a directed edge $(u, v) \in E$ if and only if $|uv| \leq r(u)$. Finally, the cost function $c : E \rightarrow \mathbb{R}_+$ represents the energy requirement for transmitting a unit size data packet along a directed edge. Following the most common theoretical models of power attenuation, the cost is taken proportional to the squared Euclidean distance:

$$c(u, v) = \begin{cases} \gamma \cdot |uv|^2 & \text{if } |uv| \leq r(u) \\ \infty & \text{else.} \end{cases}$$

\footnote{Note that $\gamma$ equals $P_{\text{rec}} / P_{\text{min}}$ from the previous chapter. For simplicity, we use $\gamma$ in this chapter. The path loss exponent $\alpha$ is set to 2.}
3.1. Introduction

The set of communication requests is modeled by a commodity set $K = \{(s_1, t_1), \ldots, (s_k, t_k)\}$, with commodity $i$ being fully described by a source device $s_i$ and a destination device $t_i$. We assume that the size of a communication request is the same for each commodity, and that no two commodities share the same source and the same destination. Hence, $k$ can be as large as $\binom{n}{2}$. If there is only one commodity, then we denote the source by $s$ and the destination by $t$. We refer to a tuple of the form $(V, E, K, p, r, c)$ as a transmission graph $T$.

A path in $T$ is a sequence of devices where any two consecutive devices share an edge. An $s-t$-path is a path that starts at device $s$ and ends at device $t$. The communication cost along an $s-t$-path $\sigma(s, t) = \{s = \sigma(1), \sigma(2), \ldots, \sigma(l(\sigma)) = t\}$, where $l(\sigma)$ is the number of devices on the path, is the sum of the transmission costs between consecutive devices on the path, i.e., equals $\sum_{i=1}^{l(\sigma)-1} c(\sigma(i), \sigma(i+1))$. A shortest path between two devices is a path with minimum transmission cost. We denote by $SP_T(s, t)$ such a shortest path between devices $s$ and $t$ in the transmission graph $T$, and by $|SP_T(s, t)|$ its cost. We suppose that every commodity in $K$ is connected by a path of finite cost, i.e., of finite transmission cost. Finally, we refer by $SP(T) = \sum_{t \in K} |SP_T(s_t, t_t)|$ to the sum of shortest path costs over all commodities.

Additional input parameters are a positive integer $\Delta n$, representing the number of additional devices $\Delta V = \{n + 1, \ldots, n + \Delta n\}$ of an agent joining $T$, and a maximal transmission range $r(v)$ for each device $v = n + 1, \ldots, n + \Delta n$.

The goal is to find optimal positions for the additional devices with respect to the objective of profit maximization for the new agent. More formally, we need to determine positions in the plane of $\Delta n$ additional devices, where device $v$ has a maximal transmission range $r(v)$, for $v \in \Delta V$, such that

$$\sum_{i=1}^{k} (SP_T(s_i, t_i) - SP_{T'}(s_i, t_i)) = SP(T) - SP(T') \quad (3.1.1)$$

is maximum, where $T'$ is the transmission graph after placing the addi-
tional devices. Alternatively, the device placement problem can be formulated as a minimization problem. Since the first term in Equation 3.1.1 is independent of the positions of the devices in $\Delta V$, the problem is equivalent to minimizing the sum of the shortest path costs over all commodities in $T'$:

$$\min SP(T'),$$  \hspace{1cm} (3.1.2)

where $T'$ again is the transmission graph including the set $\Delta V$ of additional devices. We will use both objectives in this chapter.

Depending on the form of the maximal transmission ranges of the additional devices, we study two different variants of this problem of optimally placing new devices. We use device placement problem as the overall name for both problems.

**Identical Device Placement** In an identical device placement instance, we are given a transmission graph $T = (V, E, K, p, r, c)$, a positive integer $\Delta n$, and identical maximal transmission ranges $r(n + 1) = r(n + 2) = \ldots = r(n + \Delta n)$ for the additional devices $n + 1, \ldots, n + \Delta n$. The goal is to find positions for the additional devices such that the profit is maximum.

**Individual Device Placement** In an individual device placement instance, we are given a transmission graph $T = (V, E, K, p, r, c)$, a positive integer $\Delta n$, and an individual maximal transmission range $r(v)$ for each device $v \in \Delta V$. The goal is to find positions for the additional devices such that the profit is maximum.

Clearly, the two problems do not differ for a single additional device. Hence, we simply speak in this case of the device placement problem for a single device.

### 3.1.3 Summary of Results

In the following sections, we present exact polynomial-time algorithms for some cases and $NP$-hardness proofs for other cases of device placement
3.2. Device Placement for Single Commodity

problems. Sections 3.2 and 3.3 describe the polynomial-time algorithms. We start in Section 3.2 with the case of a single additional device and a single commodity, as this provides geometric insight into the problem. In the same section, we generalize the algorithm for a single device and a single commodity to the case with multiple identical additional devices and one commodity. Next, Section 3.3 contains a polynomial-time algorithm for the case of one additional device and multiple commodities. We continue in Section 3.4 and Section 3.5 with hardness proofs. Section 3.4 contains a \( \mathcal{NP} \)-hardness proof for the general identical device placement problem, whereas Section 3.5 shows that the problem for multiple individual devices is already \( \mathcal{NP} \)-hard for two commodities. Finally, Section 3.6 provides insight on the pure shortest path problem in transmission graphs.

3.2 Device Placement for Single Commodity

This section studies the geometric basis of the device placement problem, considering first the special case with only one additional device and a single commodity, and presents a polynomial-time algorithm for this special case.

3.2.1 The Geometric Basis

We investigate the geometric basis of the device placement problem at a very elementary device configuration first, and generalize it to other configurations later. The elementary configuration consists of only two devices \( u \) and \( v \) wishing to communicate, and we are interested in the best position for an additional device \( v' \). Let the impact \( f() \) of the additional device be the difference between the cost of the direct communication from \( u \) to \( v \) and the cost of the communication from \( u \) to \( v \) via the additional device \( v' \). The following observation relates the impact to the position of the additional device \( v' \) (see Figure 3.2.1 for a plot in the plane and Figure 3.2.2 for the corresponding graph in \( \mathbb{R}^3 \)).
**Figure 3.2.1:** Points with same impact are on circles around $M(u, v)$ for the simple case with two devices $u$ and $v$.

**Observation 3.1.** The impact of an additional device $v'$ between devices $u, v$ on the source-destination pair $(u, v)$ is:

\[
\begin{align*}
    f(v', u, v, (u, v)) &= c(u, v) - (c(u, v') + c(v', v)) \\
    &= c(u, v)/2 - 2\gamma \cdot |v', M(u, v)|^2,
\end{align*}
\]

where $M(u, v)$ is the middle point of the line segment from $u$ to $v$.

**Proof.**

\[
\begin{align*}
    f(v', u, v, (u, v)) &= c(u, v) - (c(u, v') + c(v', v)) \\
    &= \frac{c(u, v)}{2} - 2\gamma \left(|u| - |v|\right)^2 + 2\left(|v'| - |Q, v|\right)^2 \\
    &= c(u, v)/2 - 2\gamma \cdot |v', M(u, v)|^2
\end{align*}
\]

Observation 3.1 implies that device positions with the same impact lie on a circle with center $M(u, v)$. The maximum impact is achieved
Figure 3.2.2: Graph of \( f(\cdot, u, v, (s, t)) \) in \( \mathbb{R}^3 \) for the simple case with two devices \( u \) and \( v \).

At \( M(u, v) \). From there the impact decreases quadratically in each direction, and it is equal to zero for positions on a circle with center \( M(u, v) \) and radius \( |uv|/2 \).

A more general configuration includes several devices, an arbitrary source-destination pair, and one additional device \( v' \). Then, the impact \( f() \) of an additional device \( v' \) on a device pair \( (u, v) \) with respect to the single source-destination pair \( (s, t) \) is defined as the difference between the shortest \( s-t \)-path without the additional device and the shortest \( s-t \)-path with additional device \( v' \) and \( (u, v', v) \) as a partial path.

Observation 3.2. The impact \( f(v', u, v, (s, t)) \) of an additional device \( v' \)
for a device pair \((u, v)\) and a source-destination pair \((s, t)\) is:

\[
f(v', u, v, (s, t)) = |SP_T(s, t)| - \left( |SP_T(s, u)| + 2 \gamma \cdot |v', M(u, v)|^2 + \frac{c(u, v)}{2} + |SP_T(v, t)| \right),
\]

where \(M(u, v)\) is the middle point of the line segment from \(u\) to \(v\).

**Proof.**

\[
f(v', u, v, (s, t)) = |SP_T(s, t)| - \left( |SP_T(s, u)| + c(u, v') + c(v', v) + |SP_T(v, t)| \right),
\]

which equals (3.2.1) using Observation 3.1. \(\square\)

Note that the impact is defined for every pair of devices \((u, v) \in V\), and that it can be negative. An additional device induces a shortest path along \((u, v', v)\) if its impact is positive. The impact is again equal for all positions with the same distance to \(M(u, v)\), and the maximum impact is achieved at position \(M(u, v)\). Observe that the circle with positions of zero impact does not necessarily go through the positions of devices \(u\) and \(v\). Indeed, if \(u\) and \(v\) are not on a shortest path before inserting the additional device, then the circle with positions of zero impact has a smaller radius than \(|M(u, v)|, u|\).

Some positions with positive impact may be unreachable due to the maximal transmission ranges of both the additional device and the existing devices. Thus, we define the profit region \(pr(u, v, (s, t))\) as the set of positions for an additional device \(v'\) where \(f(v', u, v, (s, t))\) is positive, \(u\) can reach \(v'\), and \(v'\) can reach \(v\), given the maximal transmission ranges. Geometrically, a profit region \(pr(u, v, (s, t))\) is the intersection of three disks: the disk around \(M(u, v)\) where \(f(v', u, v, (s, t)) \geq 0\), the disk with center \(u\) and radius \(r(u)\), and the disk with center \(v\) and radius \(r(v')\). See Figure 3.2.3 for three possible shapes of such an intersection. The boundary of a shape consists of at most four circle segments.
Figure 3.2.3: Profit region $pr(u, v, (s, t))$ building an asymmetric lens, a circle, and a shape bounded by four arcs.
We define $\bar{f}(\cdot)$ to be the function $f(\cdot)$ restricted to the corresponding profit region, that is, $\bar{f}(v', u, v, (s, t))$ is equal to $f(v', u, v, (s, t))$ for all positions of $v'$ inside $pr(u, v, (s, t))$ and $-\infty$ otherwise.

Assuming that the profit region $pr(u, v, (s, t))$ is not empty and that $v'$ is placed between $u$ and $v$, the position inside $pr(u, v, (s, t))$ with minimal distance to the point $M(u, v)$ is the best position for $v'$ because it maximizes $\bar{f}(v', u, v, (s, t))$. If the maximal transmission ranges of $u$ and $v'$ are large enough, then this is the same as $M(u, v)$ itself. If the maximal transmission range of $v'$ or $u$ is restricted, then the best position moves on the line segment between $u$ and $v$ towards device $u$ respectively $v$ until it enters the profit region. Such a best position is denoted by $p^*(u, v, (s, t))$, and it can be computed in constant time given the distance between $u$ and $v$ and the maximal transmission ranges $r(u)$ and $r(v')$.

The fact that an additional device reduces the cost between exactly one device pair together with the function $\bar{f}(\cdot)$ lets us state the following geometric formulation of the device placement problem for a single additional device and a single commodity:

$$\max_{(u', v') \in \mathbb{R}^2} \max_{(u, v) \in V^2} \bar{f}(v', u, v, (s, t)). \tag{3.2.2}$$

Next, we use this formulation to derive an algorithm for the single device placement problem for a single commodity.

### 3.2.2 Single Device Placement for Single Commodity

Observation 3.2 together with the formulation in (3.1.2) directly induces the following algorithm to solve the single device placement problem for a single commodity. The algorithm, called TwoLayerGraphSearch, constructs an expanded 2-layer graph to encode the restriction that only one additional device is available, and then looks for a shortest path in this graph. Figure 3.2.4 shows an expanded 2-layer graph for a simple device setting. It is constructed as follows. Each layer in the 2-layer graph contains a copy of the transmission graph. A vertex $(u, 0)$ on layer 0 corresponds to device $u$ and no additional device used on the subpath from
source $s$ to $u$, and a vertex $(u, 1)$ on layer 1 corresponds to device $u$ and one additional device used on the subpath from $s$ to $u$. Further, additional edges are introduced between layers, directed from layer 0 to layer 1. Such an edge models a transmission via an additional device. The cost of the directed edge between vertices $(u, 0)$ and $(v, 1)$, for $u, v \in V$, $u \neq v$, is equal to $c(u, p^*) + c(p^*, v)$, the transmission cost between $u$ and $v$ via an additional device at position $p^*(u, v, (s, t))$. Edges with cost infinity are not added. In this graph, we then search a shortest path from vertex $(s, 0)$ to vertex $(t, 0)$ and another one from $(s, 0)$ to vertex $(t, 1)$, and take the minimum of these two paths. Note that a shortest path does not necessarily use the additional device as the maximal transmission range of the additional device might be to small to be useful.

**Theorem 3.3.** The single device placement problem for a single commodity can be solved in time $O(n^2)$.

**Proof.** We use Algorithm TwoLayerGraphSearch. The shortest $s - t$-path using at most one additional device is the minimum of the shortest $s - t$-
Algorithm 1 Algorithm TwoLayerGraphSearch: computes optimal position for one additional device and one commodity

\begin{itemize}
\item \( G' = \text{expanded 2-layer graph} \)
\item \( SP(s, (t, 0)) = \text{shortest path from } (s, 0) \text{ to } (t, 0) \text{ in } G' \)
\item \( SP(s, (t, 1)) = \text{shortest path from } (s, 0) \text{ to } (t, 1) \text{ in } G' \)
\item \( \text{output } \min\{|SP(s, (t, 0)|, |SP(s, (t, 1))|\} \)
\end{itemize}

path using no additional device and the shortest \( s - t \)-path using exactly one additional device. The cost of a shortest path without additional device is equivalent to a shortest path from \((s, 0)\) to \((t, 0)\). The cost of a shortest path with exactly one additional device is equivalent to a shortest path from \((s, 0)\) to \((t, 1)\). This follows from Observation 3.2, implying that we can restrict our investigation to points \( p^*(u, v, (s, t)) \) for all device pairs \((u, v)\). For each position \( p^*(u, v, (s, t)) \) the algorithm computes the correct edge costs. Hence, correctness follows.

The running time includes the time to construct the expanded graph plus the time to search the shortest path in this graph. The construction of the graph is dominated by the construction of the directed edges between the layers. As the edge cost has to be computed from each vertex \((u, 0)\) on the lower level to any vertex \((v, 1)\) on the upper level, for \( u \neq v \), and the cost of such an edge can be computed in constant time, the running time for the graph construction is \( O(n^2) \). The shortest path search originating in vertex \((s, 0)\) needs \( O(n^2) \) as well with Dijkstra’s original shortest path algorithm [28].

3.2.3 Multiple Identical Device Placement for Single Commodity

Up to here, we were to add one additional device. If we are allowed to add multiple identical additional devices, then the TwoLayerGraphSearch algorithm can be extended to solve this case. In principal, the best positions for \( h \) additional devices between a fixed pair \((u, v)\) of devices are to
3.2. Device Placement for Single Commodity

distribute the additional devices in equal distances on the segment connecting $u$ and $v$, for $1 \leq h \leq \Delta n$. However, a limited maximal transmission range of device $v$ or of the additional devices may make equal distances impossible. In such a case, the additional devices are distributed on the segment connecting $u$ and $v$ as evenly as possible. The algorithm MultiLayerGraphSearch constructs a $(\Delta n + 1)$-layer graph with a copy of the transmission graph on each layer. Vertex $(u, h)$ now corresponds to device $u$ and exactly $h$ additional devices used on the subpath from $s$ to $u$, for $0 \leq h \leq \Delta n$. Between layers, there are only directed edges from a lower index layer to a higher index layer. A directed edge from $(u, h_1)$ to $(v, h_2)$ corresponds to transmitting from $u$ to $v$ via $(h_2 - h_1)$ optimally placed additional devices. Such an edge has cost equal to the sum of transmission costs along the additional devices. Edges with infinite cost are left out. In this expanded graph, we then ask for the minimum among the shortest paths between $(s, 0)$ and $(t, h)$, for $0 \leq h \leq \Delta n$.

**Algorithm 2** MultiLayerGraphSearch: computes optimal position for $\Delta n$ identical additional devices and one commodity

$G' = \text{expanded} (\Delta n + 1)$-layer graph

$SP(s, (t, h)) = \text{shortest path from} (s, 0) \text{to} (t, h) \text{in} G'$, for $0 \leq h \leq \Delta n$

output $\min_{0 \leq h \leq \Delta n} |SP(s, (t, h))|$

**Theorem 3.4.** The multiple identical device placement problem for a single commodity can be solved in time $O((\Delta n)^2 n^2)$.

**Proof.** We use Algorithm MultiLayerGraphSearch. Correctness of the algorithm can be deduced similarly to Theorem 3.3.

The running time again consists of the construction of the graph and the search for the shortest path in the expanded graph. The graph construction needs time $O((\Delta n)^2 n^2)$, as there are that many potential edges in the graph. All shortest paths can be found in time $O((\Delta n)^2 n^2)$ using Dijkstra's original shortest path algorithm [28] to find a shortest path tree rooted at $(s, 0)$.

$\square$
3.3 Single Device Placement for Multiple Commodities

With multiple commodities \(k > 1\), the optimal position for a single additional device may be different from the optimal point \(p^*(u, v, (s_i, t_i))\) between some existing devices \(u\) and \(v\), and a specific commodity \(l\). Rather, the best position could be a position where connections between several source-destination pairs use the new device. Unfortunately, the ideas from the previous section do not easily extend to a polynomial-time algorithm for the single device and multiple commodities case. Therefore, we first present a different algorithm for the single device and single commodity case, which has worse running time than the algorithm above, but is extendable to the single device and multiple commodities case.

The alternative approach to solve the single device and single commodity case is to directly use the geometric formulation in (3.2.2). To that end, we determine for each point in the plane the device pair \((u, v)\) with maximum \(\bar{f}(\cdot)\), and assign the point to this device pair. This leads to a subdivision of the plane into cells, where a cell is characterized by a device pair. Inside a cell, the best position for an additional device is equivalent to the maximum of the concave function \(\bar{f}(\cdot)\) for the characterizing device pair, and hence easy to compute. For a polynomially bounded number of cells, this approach gives rise to a polynomial time algorithm.

Before explaining the approach in more detail, we shortly recapitulate the following terms from computational geometry. We explain the terms for the 2-dimensional case as this case is most illustrative, and state the results needed later in this section for the 3-dimensional case. The definitions straightforwardly generalize to higher dimensions. We refer to a textbook as [25, 42] for an overview on computational geometry and to the book of Sharir and Agarwal [74] for a detailed exposition on arrangements and their applications.

An arrangement is the partition of the space into so called arrangement cells induced by the intersection of a collection of geometric objects. Figure 3.3.1 shows an instance with an arrangement of lines. The com-
The combinatorial complexity of an arrangement is the total number of cells in all dimensions. For instance, the combinatorial complexity in an arrangement of lines is the number of intersection points of two lines (the 0-dimensional arrangement cells), plus the number of maximal connected portions of a line (the 1-dimensional arrangement cells), plus the maximal connected regions of the plane not intersected by any line (the 2-dimensional arrangement cells). A partially defined surface in the 3-dimensional space is an object which is monotone in \( x \) and \( y \), that is, every line parallel to the \( z \)-axis intersects the surface in at most one point, and for which the domain is a closed subset of \( \mathbb{R}^2 \). The combinatorial complexity of an arrangement of \( n \) partially defined surfaces in the 3-dimensional space is \( O(n^3) \).

An arrangement possesses several substructures which are of interest by themselves. One of these substructures is the upper envelope. For the definition of the upper envelope, we see each 2-dimensional geometric object as the graph of a (partially defined) function:

\[
y = \lambda_t(x),
\]
where the domain of $x$ is a subset of $\mathbb{R}$. The upper envelope $E_\Lambda$ of a set of $n$ functions $\Lambda = \{\lambda_1, \ldots, \lambda_n\}$ is the pointwise maximum of these functions (see Figure 3.3.2):

$$E_\Lambda(x) = \max_{1 \leq i \leq n} \lambda_i(x), \text{ for all } x \in \mathbb{R}.$$ 

If $\lambda_i(x)$ is undefined for all functions $\lambda_i$, $1 \leq i \leq n$, then $E_\Lambda(x)$ is set to $-\infty$. For $n$ partially defined functions $\lambda_i(x, y)$ in the 3-dimensional space, the combinatorial complexity of the upper envelope is again the number of cells with dimension at most two, and equals $O(n^{2+\varepsilon})$, for any $\varepsilon > 0$. In a planar line arrangement such as the one in Figure 3.3.1, all functions are linear functions defined on the whole interval $\mathbb{R}$.

The maximization diagram of a set of partially defined functions is the subdivision of the line with $y = 0$ into maximal connected cells in which the upper envelope is attained by the same set of functions. Each cell is bounded by points where the upper envelope $E_\Lambda$ is attained by at least two functions or by points on the domain boundary of some $\lambda_i(x)$. We denote
3.3. Single Device Placement for Multiple Commodities

the maximization diagram of the function set \( \Lambda = \{ \lambda_1, \ldots, \lambda_n \} \) by \( M_\Lambda \). The combinatorial complexity of the maximization diagram is defined to be the same as the combinatorial complexity of the upper envelope. From an algorithmic point of view, the maximization diagram can be computed within time \( O(n^{2+\epsilon}) \) for \( n \) partially defined functions \( \lambda_i(x, y) \) in \( \mathbb{R}^3 \), for any \( \epsilon > 0 \) \cite{1}. The output is a representation where each 2-dimensional cell is stored in an ordered list of its incident edges together with the function attaining the maximum (if it exists). The maximization diagram of the instance in Figure 3.3.2 is indicated above the \( x \)-axis.

3.3.1 Single Maximization Diagram Approach

In the single device and single commodity case, the maximization diagram with the functions \( \bar{f}(\cdot) \) as set of partially defined functions is exactly the cell subdivision of the plane which assigns each point to the device pair with maximum impact. Thus, an edge in this maximization diagram arises either from the intersection of two impact functions \( \bar{f} \) or from a domain boundary of an impact function (see Figure 3.3.3 for an extract of the upper envelope and Figure 3.3.4 for the corresponding maximization diagram).

Observation 3.5. The intersection of two impact functions \( \bar{f}(\cdot) \) is a line.

Proof. Consider the impact functions for device pairs \((u_1, v_1)\) and \((u_2, v_2)\), and an additional device \( v' \). Both impact functions \( \bar{f}(v', u_i, v_i, (s, t)) \) are of the form \( H_i - 2 \cdot \gamma |v'|, M(u_i, v_i)| \), where the constant \( H_i \) depends on the positions of the devices, for \( i = 1, 2 \). If we set \( \bar{f}(v', u_1, v_1, (s, t)) = \bar{f}(v', u_2, v_2, (s, t)) \), then the set of points fulfilling the equation constitutes a line.

This observation together with the fact that the domain boundaries of all impact functions are circle segments imply that the edges of any 2-dimensional maximization diagram cell are line or circle segments. Further, each such cell is characterized either by the device pair making the largest profit for all positions inside the cell or by no device pair, if no device pair makes profit inside the cell. We call this pair the dominating pair.
Figure 3.3.3: Upper envelope of the impact functions for two device pairs.

Figure 3.3.4: Maximization diagram of the impact functions in Figure 3.3.3.
3.3. Single Device Placement for Multiple Commodities

if it exists. Next, we determine the best position inside a maximization diagram cell.

**Lemma 3.6.** Given a 2-dimensional maximization diagram cell \( c \) with a dominating pair \((u, v)\), and represented by a list of its \( n_c \) incident edges, the optimal position inside \( c \) with respect to \( \tilde{f}(\cdot, u, v, (s, t)) \) can be found in time \( O(n_c) \).

**Proof.** Inside \( c \), the profit of any position is equal to the concave impact function \( \tilde{f}(\cdot, u, v, (s, t)) \). Hence, the maximum inside \( c \) is either attained at the single point where the gradient is equal to zero, if this point lies inside \( c \), or it is attained somewhere on the boundary of \( c \). This suggests the following algorithm to compute the optimal position. We compute the single point where the gradient is zero, and check whether this point lies inside \( c \). In our case, the gradient is zero at position \( M(u, v) \). If this position is inside \( c \), we are done. Otherwise, we go along the boundary of cell \( c \). For a single edge, the maximum is attained at the position with smallest distance to \( M(u, v) \).

The running time consists of a test whether the position \( M(u, v) \) lies in \( c \) and the computation of the maximum on each edge. The test can be done in time linear in the number of edges by comparing the position to each edge. The maximum computation for all \( n_c \) edges needs linear time as well, since the position on a line segment or circle segment edge with smallest distance to \( M(u, v) \) can be determined in constant time. \( \square \)

Algorithm MaxDiagram extends the approach explained above to consider all 2-dimensional maximization diagram cells, rather than just one cell.

**Lemma 3.7.** The single device placement problem for a single commodity can be solved in time \( O(n^{4+\varepsilon}) \).

**Proof.** We use Algorithm MaxDiagram. Correctness of the algorithm follows by construction.

The for-loop over all device pairs needs time \( O(n^2) \). We first compute the single source shortest paths tree from \( s \), and the single destination
Algorithm 3 MaxDiagram: computes optimal position for one additional device and one commodity

\begin{algorithm}
\begin{algorithmic}
\ForAll {device pairs \((u, v)\)}
  \State compute \(\tilde{f}(\cdot, u, v, (s, t))\)
\EndFor
\State \(\{\tilde{P} = \cup_{u, v, \tilde{f}(\cdot, u, v, (s, t))}\}\)
\State compute maximization diagram \(M_{\tilde{P}}\)
\ForAll {2-dimensional maximization diagram cells \(c \in M_{\tilde{P}}\)}
  \State compute optimum inside \(c\)
  \State update global optimum
\EndFor
\State output global optimum
\end{algorithmic}
\end{algorithm}

shortest paths tree to \(t\). This can be done in \(O(n \log n + m)\). Next, we evaluate Equation (3.2.1) for each device pair. As a single evaluation can be executed in constant time using the shortest path trees, this step runs in time \(O(n^2)\).

The maximization diagram of the \(O(n^2)\) partially defined functions \(\tilde{f}(\cdot)\) can be computed in time \(O(n^{4+\epsilon})\). The combinatorial complexity of this maximization diagram is \(O(n^{4+\epsilon})\) as well.

The running time of the for-loop over all 2-dimensional cells in the maximization diagram consists of the computation of the maximum for each cell. Using Lemma 3.6 and the fact that each edge is incident to at most two cells, the running time of this for-loop amounts to \(O(n^{4+\epsilon})\). All together, the running time is \(O(n^{4+\epsilon})\).

\[\square\]

3.3.2 Multiple Maximization Diagrams Approach

Next, we extend the above approach to the single device placement problem for multiple commodities. To do so, we need a further concept from computational geometry. An overlay is a special form of an arrangement, namely the arrangement defined by two sets of geometric objects where
the objects within a single set are nonintersecting. The overlay of a set
$M'$ of planar geometric objects with combinatorial complexity $n_{M'}$, and
another set $M''$ of planar geometric objects with combinatorial complexity
$n_{M''}$ can be computed in time $O(n_M \log (n_{M'} + n_{M''}))$ where $n_M$ is the
combinatorial complexity of the resulting overlay (see Chapter 2 in [25]).

A geometric formulation of the single device placement problem for
multiple commodities is:

\[
\max_{(v'_i, v'_j) \in \mathbb{R}^2} \sum_{l=1}^k \max_{(u, v) \in \mathbb{R}^2} f_{v'}(u, v, (s_l, t_l)).
\] (3.3.1)

The term inside the sum corresponds to the maximization diagram for a
single commodity. Therefore, we first compute the maximization diagram
for each commodity $l \in K$, giving sets $M_{\tilde{F}_l}$ up to $M_{\tilde{F}_k}$. Now, each point
in the plane is part of one cell in each $M_{\tilde{F}_l}$, and that cell determines the
dominating pair for the corresponding commodity $l$ — if it exists. We use
this fact to determine the regions in which each point has, for each single
commodity, the same dominating pair. More precisely, we construct the
overlay $O$ of the cell sets $M_{\tilde{F}_1}, \ldots, M_{\tilde{F}_k}$. Then, for every single commod-
ity, the dominating pair is the same for every point in an overlay cell.

Lemma 3.8 states the complexity of computing the optimal position
inside an overlay cell.

**Lemma 3.8.** Given a 2-dimensional overlay cell $c$ with a (possibly empty)
dominating pair $(u_l, v_l)$ for each commodity $l \in K$, and represented by a
list of its $n_c$ incident edges, the optimal position inside $c$ with respect to the
profit $\sum_l f((u_l, v_l), (s_l, t_l))$ can be found in time $O(n_c)$.

**Proof.** The position for which $\sum_l f((u_l, v_l), (s_l, t_l))$ attains the ma-
imum is the optimal position inside $c$. As the functions $f((u_l, v_l), (s_l, t_l))$
are concave inside $c$ for all dominating pairs, and all commodities $l \in
K$, the sum over these functions is concave as well. The approach of
Lemma 3.6 carries over to this case. The maximum is either attained at
the single point where the gradient is zero, or on the boundary of $c$. Here,
Algorithm 4 MaxDiagramOverlay: computes optimal position for one additional device and multiple commodities

for all commodities \( l \) do
  for all device pairs \( (u, v) \) do
    compute \( \tilde{f}(\cdot, u, v, (s_l, t_l)) \)
  end for
  \( \{ \tilde{F}_l = \bigcup_{u, v} \tilde{f}(\cdot, u, v, (s_l, t_l)) \} \)
  compute maximization diagram \( M(\tilde{F}_l) \)
end for
compute \( O = Overlay(M_{\tilde{F}_1}, \ldots, M_{\tilde{F}_k}) \)
for all 2-dimensional overlay cells \( c \in O \) do
  compute optimum inside \( c \)
  update global optimum
end for
output global optimum

the single point where the gradient is zero evaluates to the center of mass of the positions \( M(u_l, v_l) \).

The remaining steps are the same as in Lemma 3.6. \( \square \)

Finally, we state the main algorithm of this section (see Algorithm 4).

**Theorem 3.9.** The single device placement problem for multiple commodities can be solved in time \( O(k^2 n^{8+\epsilon} \log (k n^{4+\epsilon})) \).

**Proof.** We use Algorithm MaxDiagramOverlay. The algorithm contains two for-loops with an overlay computation in between. The first for-loop over all commodities needs time \( O(k n^{4+\epsilon}) \) as we compute a maximization diagram for each of \( k \) commodities.

The combinatorial complexity of the overlay is \( O((k n^{4+\epsilon})^2) \), and can be constructed in time \( O(k^2 n^{8+\epsilon} \log (k n^{4+\epsilon})) \). In the for-loop over all overlay cells, the maximum inside each overlay cell is computed. Using Lemma 3.8 and the fact that each edge is incident to at most two cells, we yield a running time for the for-loop of \( O(k^2 n^{8+\epsilon}) \). The computation of
3.3. Single Device Placement for Multiple Commodities

\[ f(\cdot, u, v, (u, v)) = 0 \]

\[ f(\cdot, u, v, (u, v)) = \alpha_1(|uv|, c_u, c') - \alpha_2(c', c_u) \cdot |v', U(u, v)|^2 \]

**Figure 3.3.5:** Situation with a cost-of-energy value, with factors \( \alpha_1(\cdot) \) and \( \alpha_2(\cdot) \) depending on \(|uv|, c'\), and \( c_u \). The value \( c' \) respectively \( c_u \) is the cost-of-energy value of the additional device \( v' \) respectively of \( u \), and \( U(u, v) \) is the point with maximum \( f(\cdot, u, v, (u, v)) \).

the overlay dominates the overall running time of the algorithm.

3.3.3 Extensions

In Section 2.1, the concept of a cost-of-energy value was introduced. There, the cost between two devices \( u \) and \( v \) is also dependent on an internal parameter, the cost-of-energy value \( c_u \) of device \( u \). The cost-of-energy value reflects the current state of a device, e. g., the current battery level. Hence, the cost \( c(u, v) \) equals the cost-of-energy value times the proportionality factor \( \gamma \) times the squared Euclidean distance. If we allow a cost-of-energy value for each device, then the approach of subdividing the plane into cells and determining the maximum in each cell separately still works. However, the graph of the impact function \( f(\cdot) \) for two fixed devices is different (see Figure 3.3.5 for an example), as is the intersection of two impact functions.

In our analysis, we assume that the transmission costs are proportional to the squared Euclidean distance between two devices. Depending on the
environment, the distance exponent can vary between 2 and 6 [73]. Our algorithms can be adapted for the setting with an exponent different from 2, still running in polynomial time.

3.4 Multiple Identical Device Placement for Multiple Commodities

The decision version of the identical device placement problem has an additional parameter $Z$, and we ask whether there is a device placement where the sum of shortest path costs over all commodities improves by at least $Z$. More precisely, we look at the following problem:

**Problem: Identical Device Placement**

**Instance:** An instance $I = (T, \Delta n, r, Z)$ of Identical Device Placement consists of a transmission graph $T = (V, E, K, p, r, c)$, a positive integer $\Delta n$, an identical maximal transmission range $r(u)$ for each additional device $u = n + 1, \ldots, n + \Delta n$, and a positive number $Z$.

**Question:** Is there a placement for the $\Delta n$ additional devices where device $u$ has a maximal transmission range $r(u)$, for $u = n + 1, \ldots, n + \Delta n$, such that the difference $SP(T) - SP(T')$ is at least $Z$, where $T'$ is the transmission graph after the placement of the additional devices?

We prove $\mathcal{NP}$-hardness of Identical Device Placement by a reduction from a restriction of planar Exact Cover By 3-Sets (X3C). In an X3C instance $I(U, S, b)$ we are given a set $U$ of $3b$ elements, a collection of 3-element subsets $S = \{S_1, \ldots, S_{|S|}\}$ of $U$, and a budget $b$. We are looking for a subcollection of size $b$ from $S$ whose union is $U$. An X3C instance $I$ can be represented by a bipartite graph $G_I = (V, E)$ with a vertex for each element of $U$ and for each subset of $S$. When no ambiguity arises
3.4. Multiple Identical Device Placement for Multiple Commodities

we denote by $i$ (respectively $j$) the element (respectively the set) as well as the corresponding vertex in the graph. There is an edge $\{i, S_j\}$ between an element vertex $i$ and a set vertex $S_j$ if and only if $i \in S_j$. Planar X3C is the restriction of X3C in which the associated bipartite graph $G_I$ is planar, and is still $\mathcal{NP}$-complete. This result even holds for the further restriction where each element appears in either two or three sets [30]. We use this restricted version for our reduction, and denote it by X3C-3.

Before proving the hardness of IDENTICAL DEVICE PLACEMENT, we present some preliminaries used in the proof. First, we need a graph drawing result for the $\mathcal{NP}$-hardness proof.

**Lemma 3.10** ([48]). A triconnected 3-planar\(^2\) graph with $n$ vertices can be drawn with horizontal and vertical segments on an $\left[ \frac{n}{2} \right] \times \left[ \frac{n}{2} \right]$ grid such that every edge has at most one bend.

Moreover, any graph $G_I = (V, E)$ from the reduction for X3C-3 in [30] can be augmented to a triconnected 3-planar graph by adding dummy edges. The number of bends is not increased through this augmentation, and the dummy edges can be deleted again after drawing the augmented graph.

**Observation 3.11.** The algorithm from [48] to draw a 3-planar graph guarantees that from every vertex $v$, either a horizontal or vertical straight line can be drawn to the outside of the bounding box of the drawing without intersecting any other vertex. Moreover, no two such straight lines that belong to two different vertices and are orthogonal intersect each other.

There are four directions to attach a line horizontally or vertically at a vertex $v$, namely left, right, up, and down. A direction is called *free* for a vertex if an orthogonal straight line can be drawn in this direction.

As a last ingredient we show how to place devices along line segments in the plane such that certain properties hold.

\(^2\)A graph is triconnected if no removal of two vertices disconnects the graph. In a 3-planar graph, any vertex has degree at most 3.
Lemma 3.12. Given two points $p_1, p_2$ in Euclidean distance $\ell$, and $z \leq \ell^2$, we can place two devices $u, v$ at $p_1, p_2$, and further devices on the line segment between $p_1$ and $p_2$ such that $|SP(u, v)| = z$.

Proof. Let $s(p_1, p_2)$ be the line segment between $p_1$ and $p_2$. We distinguish three cases. If $z = \ell^2$, then we place device $u$ with $r(u) = \ell$ at position $p_1$ and $v$ at position $p_2$, and no other device on $s(p_1, p_2)$. If $z$ is a multiple of $\ell$, then we place device $u$ at $p_1$ and $v$ at $p_2$, plus devices on $s(p_1, p_2)$ at distance $h \cdot z/\ell$ from $p_1$, for $h = 1, \ldots, \ell^2/z - 1$. The maximal transmission range of all devices is set to $z/\ell$.

Otherwise (see Figure 3.4.1), we place device $u$ at $p_1$ and $v$ at $p_2$, plus devices on $s(p_1, p_2)$ at distance $h \cdot z/\ell$ from $p_1$, for $h = 1, \ldots, \lfloor \ell^2/z \rfloor - 2$. Let $r_\ell = \ell - \lfloor \ell^2/z \rfloor \cdot z/\ell$. A further device is placed on $s(p_1, p_2)$ at distance $\ell - r_\ell + a$ from $p_1$, where $a = (\sqrt{(z/\ell)^2 - r_\ell^2} - z/\ell + r_\ell)/2$. The maximal transmission range of the devices are set such that each device reaches the next device on the segment. The shortest path between $u$ and $v$ goes over each device on $s(p_1, p_2)$. The sum of the squared distances is $(\lfloor \ell^2/z \rfloor - 2)(z/\ell)^2 + (z/\ell + a)^2 + (r_\ell - a)^2$, which equals $z$. \qed

For ease of description, $ch(p_1, p_2, z)$ denotes a chain of devices between positions $p_1$ and $p_2$, with positions as in the proof above, and a path of cost $z$. Similarly, $ch^{-}(p_1, p_2, z)$ is the same set of devices without the device at position $p_1$. 

Figure 3.4.1: Chain construction.
3.4. Multiple Identical Device Placement for Multiple Commodities

The idea of the reduction is to introduce a device for every element and every set from the X3C-3 instance plus a global destination device (see Figure 3.4.2). Every element device forms one source-destination pair with the global destination device. Chains of devices ensure that the only possible paths between an element device and the global destination device go through the set devices the element is member of. The cost of these paths is the same for all source-destination pairs. Moreover, consecutive devices on the chains are so close that an additional device in between does not yield a large reduction on the path cost. There is only one position inducing a large improvement between each set device and the global destination device where consecutive devices are far apart. Hence, the number of reasonable positions for the additional devices is limited to the number of subsets in $S$, and there is a one-to-one correspondence between such a position and a subset.

Theorem 3.13. IDENTICAL DEVICE PLACEMENT is $\mathcal{NP}$-hard.

Proof. The reduction is from planar X3C-3. Let $I(U, S, b)$ be an instance of planar X3C-3. By Lemma 3.10, we can assume that an orthogonal grid embedding is given, for which we scale all coordinates by a factor $\hat{c}$ to ensure that vertices are sufficiently far away from each other. The factor $\hat{c}$ is polynomially bounded in $n$, and will be defined precisely later. After the scaling, a vertex $i$ has coordinates $(i_x, i_y)$. Also by Lemma 3.10, each edge $(i, j)$ between vertex $i$ and $j$ in the embedding consists of at most two line segments, and we denote the line segment of edge $(i, j)$ connected to element vertex $i$ by $s_i(i, j)$, and the line segment connected to set vertex $j$ by $s_j(i, j)$. If only one line segment builds the edge, then we split the line segment into two line segments of equal length and assign those to $s_i(i, j)$ respectively $s_j(i, j)$. By $l(\cdot)$ we refer to the Euclidean length of an edge respectively a line segment. Moreover, we let $f_i(i, j) = 1/l(s_i(i, j))$ and $f_j(i, j) = 1/l(s_j(i, j))$.

We construct instance $\hat{I}(T, \Delta n, \tau, Z)$ of IDENTICAL DEVICE PLACEMENT from $I$ as follows. We start with the description of the positions and transmission ranges of the devices in the transmission graph $T$. The device set $V$ contains five different kinds of devices, called the element, the
Figure 3.4.2: Overview illustration of the reduction where each $v_i^e$ corresponds to an element device, and each $v_j^s$ to a set device. $v_i^T$ is the global destination device, and each $V_i^{i,j}$ respectively each $V_j^{j,T}$ refers to an element-set chain respectively a set-destination chain. The gaps on the set-destination chains indicate the potential positions for the additional devices.

set, the global destination, the element-set chain, and the set-destination chain devices. We describe each kind separately (see Figure 3.4.2 for an overview illustration).
3.4. Multiple Identical Device Placement for Multiple Commodities

Element devices

For each element \( i \in U = \{1, \ldots, |U|\} \), we place an element device \( v^e_i \) at position \( (i_x, i_y) \). The maximal transmission range of such a device is set to \( \max_{j: (i, j) \in E} 3/2 \cdot f_i(i, j) \). We denote the union of all element devices by \( V^e \).

Set devices

For each set \( j \in S = \{1, \ldots, |S|\} \), we place a set device \( v^s_j \) at position \( (j_x, j_y) \). These devices have a maximal transmission range equal to 2. We denote the union of all set devices by \( V^s \).

Global destination device

We place one single global destination device \( v^T \) at position \( (0, -M) \), where \( M \) is a large number to be defined later. The maximal transmission range of this device is set arbitrarily.

Element-set chain devices

These devices model the connection between an element and the set the element is a member of. For each element \( i \) and set \( j \) for which \( i \in S_j \), we introduce an element-set chain \( V^{i,j} \). The element-set chain \( V^{i,j} \) looks as follows: we place devices \( V^{i,j} = \{v^{i,j}_1, \ldots, v^{i,j}_{n_{i,j}}\} \), with \( n_{i,j} = l(s_i(i, j))^2 + l(s_j(i, j))^2 \), along the edges of the embedding (dotted line segments in the overview figure). The placement is done such that the distance between two consecutive devices on a line segment is at most 1, and the cost of a path from an element device to any of the at most three set devices is the same. To achieve this (see Figure 3.4.3), we place a first device \( v^{i,j}_1 \) at distance \( 3/2 \cdot f_i(i, j) \) from the element device \( v^e_i \). The maximal transmission range of this device is set to \( 1/2 \cdot f_i(i, j) \). Next, we place three devices \( v^{i,j}_h \) for \( h = 2, \ldots, 4 \) at distance \( 3/2 \cdot f_i(i, j) + (h - 1)/2 \cdot f_i(i, j) \)
from the element device $v_i^e$. Devices $v_{2i}^{i,j}$, $v_{3i}^{i,j}$ have a maximal transmission range equal to $1/2 \cdot f_i(i,j)$, and $v_{4i}^{i,j}$ one of $f_i(i,j)$. Devices $v_{h}^{i,j}$ for $h = 5, \ldots, l(s_i(i,j))^2$ with a maximal transmission range $f_i(i,j)$ are positioned at distances $(h - 1) \cdot f_i(i,j)$ from the element device $v_i^e$. At the common endpoint of the two line segments $s_i(i,j)$ and $s_j(i,j)$, a further device $v_{l(s_i(i,j))^2+1}^{i,j}$ with a maximal transmission range $f_j(i,j)$ is placed. We continue with devices on line segment $s_j(i,j)$. We uniformly distribute devices on this line segment by placing devices $v_{l(s_i(i,j))^2+1+h}^{i,j}$ at distances $h \cdot f_j(i,j)$ from the common endpoint of $s_i(i,j)$ and $s_j(i,j)$, for $h = 1, \ldots, l(s_j(i,j))^2 - 1$. All devices on $s_j(i,j)$ have a maximal transmission range of $f_j(i,j)$. By the above construction, there is a path from any element device to a set device through the devices $(v_1^{i,j}, \ldots, v_{l(s_i(i,j))^2}^{i,j})$ with cost 2. Moreover, a path in the reverse direction is not possible due to the chosen maximal transmission ranges.
3.4. Multiple Identical Device Placement for Multiple Commodities

Figure 3.4.4: The black dots on x-coordinate equal to ć correspond to the devices of a set-destination chain $V^j,T$ from set device $v_j^S$ to device $v_j^{UG}$. The circles are devices belonging to element-set chains.
Set-destination chain devices

These devices connect each set device to the global destination device. The (unavoidable) crossings with the element-set chains have to be constructed carefully for the reduction to work. Moreover, we have to take care that no two set-destination chains cross. There is a set-destination chain \( V^j \) from each set device \( v^j \) to the global destination device \( v^T \), for \( j \in \{1, \ldots, |S|\} \) (dashed lines in the overview figure). In detail, the following devices are set. Let \( W = \lceil \frac{n}{2} \rceil \). We distinguish three cases depending on the free direction of the set vertex \( j \). Note that the graph drawing [48] does not contain a vertex where the free direction of a set vertex is upwards.

**Free direction of set vertex \( j \) is downwards** We place chains of devices interrupted by longer distances without devices in the free direction (see Figure 3.4.4 for a schematic illustration). More precisely, we place \((W - 1)\) chains \( ch((j_x, j_y - h \cdot \hat{c} - 2), (j_x, j_y - (h + 1)\hat{c} + 2), 1) \), where \( h = 0, \ldots, W - 2 \). The devices have a maximal transmission range of \( 1/(\hat{c} - 4) \) except the last device on each chain, which has a maximal transmission range of \( 4 \). Finally, we place one more chain \( ch^-((j_x, j_y - (W - 1)\hat{c} + 2), (j_x, -(W + 1)\hat{c}), 4) \). We denote the “last” device, at position \((j_x, -(W - 1)\hat{c})\), by \( v^{UG}_j \). The maximal transmission range of these devices except \( v^{UG}_j \) is set such that any device can reach the next device on the chain, and \( v^{UG}_j \)'s maximal transmission range equals \( d_{gap} \), which will be defined later.

**Free direction of set vertex \( j \) is to the left** Again, we place chains of devices interrupted by longer distances without devices in the free direction. More precisely, we place \((W - 1)\) chains \( ch((j_x - h \cdot \hat{c} - 2, j_y), (j_x - (h + 1)\hat{c} + 2, j_y), 1) \) for \( h = 0, \ldots, W - 2 \). The devices in the chains have a maximal transmission range of \( 1/(\hat{c} - 4) \), except the last device on each chain, having a maximal transmission range of \( 4 \). Further, we add a chain \( ch^-((j_x - (W - 1)\hat{c} + 2, j_y), -(W - 1)\hat{c} - j_y, 2) \) and a chain \( ch^-((- (W - 1)\hat{c} - j_y, j_y), -(W - 1)\hat{c} - j_y, -(W - 1)\hat{c}), 2) \). We denote the device at position \((- (W - 1)\hat{c} - j_y, -(W - 1)\hat{c})\) by \( v^{UG}_j \). The
3.4. Multiple Identical Device Placement for Multiple Commodities

maximal transmission range of the devices in the latter two chains except \(v_{j}^{UG}\) is set such that any device can reach the next device on the chain, and \(v_{j}^{UG}\)'s maximal transmission range equals \(d_{gap}\).

**Free direction of set vertex \(j\) is to the right**  This case is similar to the previous case except that the chain first moves in the other direction. More precisely, we place \((W - 1)\) chains \(ch(\langle j_{x} + h \cdot \hat{c} + 2, j_{y} \rangle, \langle j_{x} + (h + 1) \hat{c} - 2, j_{y} \rangle, 1)\) for \(h = 0, \ldots, W - 2\). The devices have a maximal transmission range of \(1/(\hat{c} - 4)\), except the last device on each chain, having a maximal transmission range of 4. Further, we add a chain \(ch^{-}(\langle j_{x} + (W - 1) \hat{c} + 2, j_{y} \rangle, \langle \hat{c}W + (W - 1) \hat{c} + j_{y}, j_{y} \rangle, 2),\) and a chain \(ch^{-}(\langle \hat{c}W + (W - 1) \hat{c} + j_{y}, - (W - 1) \hat{c} \rangle, 2)\). We denote the device at position \(\langle \hat{c}W + (W - 1) \hat{c} + j_{y}, - (W - 1) \hat{c} \rangle\) by \(v_{j}^{UG}\). The maximal transmission range of the devices in the latter two chains except \(v_{j}^{UG}\) is set such that any device can reach the next device on the chain, and \(v_{j}^{UG}\)'s maximal transmission range equals \(d_{gap}\).

The construction of the set-destination chains yields a device \(v_{j}^{UG}\) for every set device \(v_{j}^{s}\) on the horizontal line \(y = -(W - 1) \hat{c}\). Moreover, there exists a path between any set device \(v_{j}^{s}\) and the corresponding device \(v_{j}^{UG}\).
with cost $2^2 + (W - 1) \cdot 1 + (W - 2) \cdot 4^2 + 4 = 17W - 25$.

As a next step in the construction of the set-destination chains, we build a set of devices resembling a tree. We place devices on the $h = \lceil \log |S| \rceil$ consecutive integer $y$-coordinates $y_\ell = -(W - 1)c - d_{gap} - \ell$, for $\ell = 0, \ldots, h$. We refer to these $y$-coordinate as levels, where level $0$ is the level with the largest $y$-coordinate. The number of devices placed on level $\ell$ equals $n_\ell = \lfloor |S|/2^\ell \rfloor$. On level $0$, the $n_0$ devices have the same $x$-coordinate as the devices $v_j^{UG}$. We call these devices $v_j^{LG}$, and refer to the gap between $v_j^{UG}$ and $v_j^{LG}$ as the large gap. On level $1$, we compute the $x$-coordinates from two devices of level $0$. We start at the left side. We take the $x$-coordinate of the two devices with smallest $x$-coordinate from level $0$, calculate the median of their $x$-coordinate and position a device on this median on level $1$. We continuously repeat this procedure with the two devices with next largest $x$-coordinate from level $0$ until no two devices are left. If there is one device left, then we place a device with same $x$-coordinate on level $1$. We iteratively continue this procedure on the other levels until $\ell = h$. The maximal transmission range for all these devices is set to $1/2$, except for the rightmost device on a level with an odd number of devices, which has maximal transmission range $1$. Between the levels, we add further devices: for each pair of devices $u, v$, which together calculate a median, we add a chain $ch(\langle u_x, u_y - 1/2 \rangle, \langle (u_x + v_x)/2, u_y - 1/2 \rangle, 1/2)$ and a chain $ch^-(\langle v_x, v_y - 1/2 \rangle, \langle (u_x + v_x)/2, v_y - 1/2 \rangle, 1/2)$ with maximal transmission ranges set such that any device can reach the next device on its chain. The device at position $\langle (u_x + v_x)/2, u_y - 1/2 \rangle$ has a maximal transmission range of $1/2$. To complete the set-destination chains, two chains of devices are placed between the single device on level $h$ and the global destination device $v_T$ with a path of cost 2, one chain on the vertical line segment down to $y$-coordinate equal to $-M$, and one chain on the horizontal line towards $v_T$, with maximal transmission ranges sufficiently large to reach the next device towards the global destination device.

The construction of the set-destination chains leads to a path between any element device and the global destination device $v_T$ with total cost of $17W - 21 + (d_{gap})^2 + \lceil \log S \rceil$. Depending on whether the element is member of two or three sets there are two or three paths with this cost per
element device. The total number of devices in the instance \( \hat{I} \) is polynomial in the input parameters.

As a next step in the reduction, we describe the commodity set \( K \) of the transmission graph. We construct \( k = |U| \) commodities with each element device \( v^e_i, i = 1, \ldots, |U| \), being a source, and device \( v^T \) being the global destination device for each such source. Without additional devices, a shortest path for any commodity goes from the source over an element-set chain to a set device that the source is member of, and from there along a set-destination chain to the global destination device.

Finally, we set \( \Delta n = b \) and \( Z = |U| \cdot (d_{gap})^2 \). The maximal transmission range \( r \) for the additional devices is set to \( d_{gap}/2 \).

Further, we set the scaling factor \( \hat{c} \) to \( 4n^3 \), the distance of the gap \( d_{gap} \) to \( n^2 \), and the \( y \)-coordinate \(-M\) of the global destination device to \(-n^4\). This completes the construction, which is computable in polynomial time.

To complete the reduction we show that \( I \) is a YES-instance of X3C-3 if and only if \( \hat{I} \) is a YES-instance of DEVICE PLACEMENT. We first show the forward-direction, that is, given a cover of size \( b \) we can construct a device placement with an improvement \( Z \). Let the indices of the \( b \) sets in the X3C-3 solution be \( \{sol_1, \ldots, sol_b\} \). Then, consider the placement of \( \Delta n \) additional devices at positions \( \big((v^U_{sol_i})_x, (v^U_{sol_i})_y - d_{gap}/2\big) \) for \( i = 1, \ldots, \Delta n \), i.e. in the middle of the large gap on the set-destination chains. For each commodity \( j \), the cost between \( v^U_j \) and \( v^L_j \) is halved, resulting in a total improvement \( Z = |U| \cdot (d_{gap})^2 \).

Next, we show the reverse direction by showing that if there is no cover of size \( b \) then there is no device placement with an improvement \( Z \). Consider any device placement in \( \hat{I} \). If we place all additional devices in the large gap, then there are source-destination paths with no improvement, resulting in total improvement less than \( Z \). If we place none of the additional devices in the large gap, then there is no other position where a large enough improvement can be attained. The last option is to place some of the additional devices in the large gap and the remaining additional devices somewhere else. These remaining devices can either be positioned on an existing path or induce a new path. Any placement on an existing
Chapter 3. Strategic Placement of Devices

path does not yield enough improvement. A new path can be induced at two positions, either by placing the additional device on an existing device \(v_{i,j}^l\), or by placing it near an element-set chain at the position where a set-destination chain crosses it. Due to the larger maximal transmission range of the additional device, a new path is induced for one commodity, which yields an improvement of at most \(2 + 16(W - 2) + (W - 1)\) in addition to the improvement from the large gap. Let \(g\) be the number of commodities for which a new path is induced. Then \(k - g\) commodities use an existing path. To let every pair use an additional device over the large gap, \(\frac{k - g}{3} + g = \frac{k}{3} + \frac{2}{3}g > \Delta n\) additional devices would be necessary. Thus, there exists a path for some commodity not going via an additional device over the large gap. The total additional improvement from the newly induced paths does not add up to \((d_{gap})^2/2\), resulting in a total improvement less than \(Z\).

Our proof also works for other metrics as for instance the Manhattan or the Chebyshev metric. To see this, note that the reduction only makes use of horizontally or vertically connected devices.

3.5 Individual Device Placement

Up to here, this chapter focused on the identical device placement problem where each device has the same maximal transmission range. Therefore, it did not play a role where we placed which additional device. In this section, we look more closely at the individual device placement where each additional device has a specific maximal transmission range. Consequently, we have to specify exactly which additional device is placed at which position. Naturally, the individual device placement only differs from the identical device placement for \(\Delta n\) larger than one. The decision version of the individual device placement problem is defined as follows:
3.5. Individual Device Placement

**Problem: Individual Device Placement**

**Instance:** An instance \( I = (T, \Delta n, r, Z) \) of **Individual Device Placement** consists of a transmission graph \( T = (V, E, K, p, r, c) \), a positive integer \( \Delta n \), an individual maximal transmission range \( r(u) \) for each additional device \( u \), for \( u = n + 1, \ldots, n + \Delta n \), and a positive number \( Z \).

**Question:** Is there a placement for the \( \Delta n \) additional devices where device \( u \) has maximal transmission range \( r(u) \), for \( u = n + 1, \ldots, n + \Delta n \), such that \( SP(T') \) is less or equal to \( Z \), where \( T' \) is the transmission graph after the placement of the additional devices?

Since the identical device placement problem is a special case of the individual device placement problem, we can immediately state the following corollary:

**Corollary 3.14. Individual Device Placement is \( \mathcal{NP} \)-hard.**

Next, we prove that the individual device placement problem is already \( \mathcal{NP} \)-hard for only two commodities.

**Theorem 3.15. Individual Device Placement for Two Commodities is \( \mathcal{NP} \)-hard.**

**Proof.** The proof is by a reduction from the partition problem, which is \( \mathcal{NP} \)-complete (see SP12 in [38]). In **Partition**, we are given a set \( A = \{a_1, \ldots, a_{|A|}\} \) of positive integer numbers with \( B = \sum_{a_i \in A} a_i \). The goal is to decide whether there is a subset \( A' \subseteq A \) such that \( \sum_{a_i \in A'} a_i = B/2 \). Consider the device placement instance specified by the four devices \( \{1, \ldots, 4\} \) in Figure 3.5.1. Device 1 is at position \((0, 0)\), device 2 at position \((B/2 + 1, 0)\), device 3 at position \((0, M)\), and device 4 at position \((B/2 + 1, M)\), with \( M \) an integer much larger than \( B \). The maximal transmission ranges of these devices are set to one. Further, the device pair \((1, 2)\) constitutes the first commodity, and the device pair \((3, 4)\) the other one. The number \( \Delta n \) of additional devices is set to \( |A| \), and the maximal transmission range \( r(u) \) is set to \( a_{u-n} \), for \( u = n + 1, \ldots, n + \Delta n \).
Finally, we set $Z$ to $2 + \sum_{a_i \in A} a_i^2$. A solution of the partition problem immediately gives a solution for the device placement problem: we place the devices with index in $A'$ one after another on the line segment between $s_1$ and $t_1$, starting at distance 1 from $s_1$, such that their maximal transmission ranges are just exactly large enough to reach the next device. The remaining devices in $A \setminus A'$ are placed between $s_2$ and $t_2$ in a similar fashion. The total cost of the shortest path for the first commodity is now equal to $1 + \sum_{a_i \in A'} a_i^2$, and that for the second commodity is equal to $1 + \sum_{a_j \in A \setminus A'} a_j^2$. If the partition problem has no solution, then no placement of the devices connects both source-destination pairs, and total shortest path cost of infinity cannot be avoided.

The status of the multiple individual device placement problem for a single commodity is still open. This is in contrast to the multiple identical device placement for a single commodity, for which we presented a polynomial-time algorithm in Section 3.2.3. However, the expanded layer
3.6. Shortest Paths in Transmission Graphs

The investigations in the previous sections only exploited the geometric structure of the problem to a small extent, for determining the optimal position between two existing devices. Other than that, the algorithms did not incorporate the fact that the cost between devices are connected to the geometric positions of the devices. To gain more insight into the geometry, we study the pure shortest path problem and the shortest path problem over a restricted number of hops in transmission graphs. We focus on two known approaches to each of this problem.

3.6.1 Sparse Distance Preserving Subgraphs

The most fundamental shortest path problem in a transmission graph \( T = (V, E, K, p, r, c) \) has \(|K| = 1\) and asks for a shortest path between a single source \( s \) and a single destination \( t \). As mentioned in Section 3.1.1, we find such a path in time \( O(m + n \log n) \) [36] using Dijkstra's shortest path algorithm implemented with Fibonacci heaps. The running time of this particular algorithm can be quadratic in the number of vertices achieved by a transmission graph where \((V, E')\) constitutes a complete graph. Since the costs \( c(u, v) = |uv|^2 \) are related to geometric positions of the devices, the question arises whether it is possible to exploit this geometric structure.

One idea to do so is to determine a subgraph of the transmission graph consisting of all vertices and a subset of the edges such that the subgraph

1. is sparser than the transmission graph (ideally, the number of edges in the subgraph is linear in \( n \)), and

2. contains a shortest path between each pair of vertices.
Subsequently, we use Dijkstra’s algorithm to find a shortest path in this subgraph. For shortest path problems in a similar setting, many researchers have taken concepts from geometric proximity structures to derive sparse subgraphs. One such geometric proximity structure is the Gabriel graph, defined as follows. Let $\text{disk}(p, q)$ be the disk (the boundary and its interior) with diameter $|pq|$ going through $p$ and $q$. The Gabriel graph $GG(V)$ is an undirected graph defined on a point set $V$ and contains an edge $(u, v)$ for $u, v \in V$ if and only if $\text{disk}(u, v)$ contains no other point from $V$ [37]. It is well known that the Gabriel graph $GG(V)$ is a planar subgraph of the complete graph on the point set $V$, and that it contains a shortest path between each pair of vertices if the cost between any two points is its squared Euclidean distance (see for instance [51]). Since a planar graph has at most $3n - 6$ edges, a shortest path in a transmission graph can be found in time $O(n \log n)$ if the transmission ranges are so large that each device can reach any other device. Other geometric proximity structures such as the relative neighborhood graph [79] or the Delaunay triangulation have similar properties, and have been suggested as sparse subgraphs as well.

Note that, using this definition, the Gabriel graph for a point set $V$ from a transmission graph $T$ may contain an edge $(u, v)$ that is longer than the maximal transmission range of $u$. To avoid such a situation, extensions of the original Gabriel graph have been considered. We first present the extension on a special class of transmission graphs, the unit disk graphs, and afterwards discuss the compatibility with more general classes of transmission graphs. The unit disk graph $UDG(V)$ for a point set $V$ is the undirected graph with an edge $(u, v)$ if and only if $|uv|$ is at most one. The unit disk graph is a commonly used simple model of an ad-hoc network, and can be seen as the transmission graph when all devices have the same maximal transmission range of one.\footnote{If all devices have the same maximal transmission range, we can take the maximal transmission range as our unit distance.} Consider the intersection $UDGG(V) = GG(V) \cap UDG(V)$ between the Gabriel graph $GG(V)$ and the unit disk graph $UDG(V)$. It can be shown that $UDGG(V)$ preserves shortest paths between vertex pairs from $UDG(V)$ when the edge
3.6. Shortest Paths in Transmission Graphs

Figure 3.6.1: A device configuration where the dotted circles specify the transmission ranges, its mutual inclusion graph, and the Gabriel graph of the same point set.

costs are equal to the squared Euclidean distance. Since $UDDG(V)$ is a subgraph of $GG(V)$, it is planar as well and allows shortest paths search in time $O(n \log n)$.

In a general transmission graph, where the devices do not have the same maximal transmission range, the edges are directed, and it is not clear how the Gabriel graph of a point set could take directions into consideration. Even if we consider a concept of symmetry, the Gabriel graph is problematic for our purposes. Indeed, the mutual inclusion graph [49] is the transmission graph where an undirected edge $(u, v)$ exists if and only if $u$ can reach $v$ and $v$ can reach $u$ as well. The instance in Figure 3.6.1 shows that the Gabriel graph cannot be used as a sparse subgraph for the mutual inclusion graph.

Consequently, geometric proximity structures only yield a subgraph fulfilling the two requirements listed above when all maximal transmission ranges are either equal or very large. For arbitrary maximal transmission ranges, we next show a negative result. The result states that the two requirements are not simultaneously attainable by providing an instance in which the total number of edges is $\Theta(n^2)$, and in which every edge is part of some shortest path.
Figure 3.6.2: A device configuration where the number of edges that are part of a shortest path is quadratic in the number of devices.

The instance is illustrated in Figure 3.6.2, and has \( n/2 \) devices on the left side, evenly distributed between position \( (-1, 0) \) and the origin \( (0, 0) \). We denote them by \( v_1^L, \ldots, v_{n/2}^L \) starting at the left most device. These devices all have a very limited maximal transmission range and cannot reach any other device. The other \( n/2 \) devices are placed on the right side, and their maximal transmission ranges allow them to reach all other devices. Hence, each device on the left side has no outgoing edge, and each device on the right side has a directed edge to all other devices in the transmission graph. The exact positions of the devices on the right side determine whether such an edge is part of a shortest path and are as follows.

We place a first device \( v_1^R \) at position \( (n, -n) \). Let \( l_1 \) be the line through \( v_1^L \) and \( v_1^R \), \( l_1^L \) the line perpendicular to \( l_1 \) going through point \( v_1^R \), and \( l_1^L \) the line parallel to \( l_1 \) through the point \( v_{n/2}^L \). We place the second device \( v_2^R \) at the intersection of \( l_1^L \) and \( l_1^L \). The same construction is repeated with point \( v_2^R \). Let \( l_2 \) be the line through \( v_2^L \) and \( v_2^R \), \( l_2^L \) the line perpendicular to \( l_2 \) going through point \( v_2^R \), and \( l_2^L \) the line parallel to \( l_2 \) through the point \( v_{n/2}^L \). We place the third device \( v_3^R \) at the intersection of \( l_2^L \) and \( l_2^L \). The
construction continues in the same way until \( n/2 \) devices are placed on the right side. Note that by the law of cosine, the edges \((v^R_{i_1}, v^L_{j})\) and \((v^R_{i_2}, v^L_{j})\) are simultaneously a shortest path edge from \( v^R_{i_1} \) respectively \( v^R_{i_2} \) to \( v^L_{j} \) if the angle \( \angle v^R_{i_1} v^R_{i_2} v^L_{j} \) as well as the angle \( \angle v^R_{i_2} v^R_{i_1} v^L_{j} \) is at most 90°, for \( 1 \leq i_1, i_2, j \leq n/2 \), and \( i_1 \neq i_2 \). The following observations guarantee that any edge from a device on the right side to a device on the left side constitutes a shortest path. The first two observations in Observation 3.16 allow us to only look at the two devices \( v^L_{1} \) and \( v^L_{n/2} \) for any pair of devices on the right side. Observations 3.16.3 and 3.16.4 show that the angle is at most 90° for any pair of devices on the right side and \( v^L_{1} \) respectively \( v^L_{n/2} \).

**Observation 3.16.** The above construction guarantees that

1. the angle \( \angle v^R_{i_2} v^R_{i_1} v^L_{j} \) is less than \( \angle v^R_{i_2} v^R_{i_1} v^L_{j} \), for any \( 1 \leq i_1 < i_2 \leq n/2 \), and for any \( j \in \{2, \ldots, n/2\} \).

2. the angle \( \angle v^R_{i_2} v^R_{i_1} v^L_{j} \) is less than \( \angle v^R_{i_2} v^R_{i_1} v^L_{j} \), for any \( 1 \leq i_1 < i_2 \leq n/2 \), and for any \( j \in \{1, \ldots, n/2 - 1\} \).

3. the angle \( \angle v^R_{i_2} v^R_{i_1} v^L_{j} \) is at most 90°, for any \( 1 \leq i_1 < i_2 \leq n/2 \). It is equal for \( i_2 = i_1 + 1 \) and less for \( i_2 > i_1 \).

4. the angle \( \angle v^R_{i_2} v^R_{i_1} v^L_{j} \) is at most 90°, for any \( 1 \leq i_1 < i_2 \leq n/2 \). It is equal for \( i_2 = i_1 + 1 \) and less for \( i_2 > i_1 \).

The total number of edges in the instance equals \( n/2 \cdot n/2 \). Therefore, the approach to run Dijkstra's algorithm on a suitable subgraph of the transmission graph does not generally accelerate the shortest path search in transmission graphs.

### 3.6.2 \( \kappa \)-hops Shortest Path

Next, we study the problem of finding a shortest \( s-t \)-path using a restricted number of hops (edges) in a transmission graph \( T = (V, E, K, p, r, c) \). The motivation to investigate this problem is twofold. One the one hand, the problem and the solution method will reappear in Section 4.2. On the other
hand, it is a nice example of a transformation into a geometric optimization problem. Similar ideas area probably applicable to the pure shortest path problem in transmission graphs. Letting $\kappa$ be the maximal allowed number of hops, we ask for a $\kappa$-hops shortest $s - t$-path. The basic approach in this section stems from the work in [11], who solved the problem for the case where the transmission graph is a complete graph, that is, each device has a maximal transmission range large enough to reach any other device. We sketch some implications for the case where the transmission graph is not complete.

To start with, we present a dynamic programming approach in Algorithm 5 which is essentially the Bellman-Ford shortest path algorithm with an adapted outer for-loop. It is based on the fact that a $i$-hops shortest path from the source to a vertex $v$ is either the $(i - 1)$-hops shortest path to $v$ or the $(i - 1)$-hops shortest path to a vertex $\bar{v}$ adjacent to $v$ together with the edge $(\bar{v}, v)$. This leads to the following recursive formula:

$$M[v, i] = \min\{M[v, i - 1], \min_{(v, \bar{v}) \in E}(M[v, i - 1] + c(\bar{v}, v))\} \quad (3.6.1)$$

where $M[v, i]$ denotes the $i$-hops shortest path from source $s$ to $v$. We stop the outer for-loop after $\kappa$ iterations in contrast to the classical Bellman-Ford algorithm, which lets the counter go up to $n - 1$ (see Algorithm RestrictedBellmanFord).

**Algorithm 5 RestrictedBellmanFord: computes $\kappa$-hops shortest $s - t$-path**

1: $\{M[v, i]\}$ stores lowest cost from source $s$ to vertex $v$ using at most $i$ edges
2: $M[s, i] = 0$ and $M[v, i] = \infty$ for all other $v \in V$ and all $i \in [0, \kappa]$
3: for all $i = 1, \ldots, \kappa$ do
4: for all $v \in V$ do
5: $M[v, i] = \min\{M[v, i - 1], \min_{(v, \bar{v}) \in E}(M[\bar{v}, i - 1] + c(\bar{v}, v))\}$
6: end for
7: end for
8: output $M[t, \kappa]$
Lemma 3.17. The $\kappa$-hops shortest $s - t$-path problem can be solved in time $O(\kappa mn)$.

Proof. We use Algorithm RestrictedBellmanFord. Due to the invariant that $M[v, i]$ stores the lowest cost from source $s$ to $v$ using at most $i$ edges, the algorithm computes the correct result.

The computation of an entry $M[v, i]$ takes time $O(d_{in}(v))$ where $d_{in}(v)$ is the indegree of vertex $v$. Hence, a single iteration over all vertices needs time $O(m)$. Since the outer-loop is executed $\kappa$ times, the total running time adds up to $O(\kappa m)$. \qed

Algorithm RestrictedBellmanFord investigates each edge in each single iteration, and thus needs time linear in the number of edges per iteration. If each device has an unrestricted maximal transmission range, [11] use the geometric structure of the problem to speed up such an iteration. So, we assume that each device has a transmission range large enough to reach any other device. Let us consider the expression (3.6.1) for a fixed device $u$ and iteration $i$. We transform the expression into a distance problem in the 3-dimensional space. First, the position of device $u$ is embedded into the plane with $z = 0$, to $p_u = (u_x, u_y, 0)$. Further, we encode the cost $M[v, i - 1]$ from source device $s$ to device $v$ using at most $i - 1$ hops in the $z$-coordinate as follows. Let $f_i(v) : V \rightarrow \mathbb{R}^3$ be defined by $f_i = (v_x, v_y, \sqrt{M[v, i - 1]})$, for each $v \in V$, resulting in a set of $n$ points $F_i(V) = \cap_{v \in V} f_i(v)$ (see Figure 3.6.3). By the Pythagorean theorem, for each $v \in V$, it holds that the squared Euclidean distance from $f_i(v)$ to point $p_u$ is

$$|f_i(v), p_u|^2 = |p_v, p_u|^2 + \sqrt{M[v, i - 1]}^2 = M[v, i - 1] + c(v, u).$$

So, we reduced the evaluation of expression (3.6.1) to the following 3-dimensional nearest neighbor problem: Given $n$ points $F_i(V)$ and a query point $p_u$, determine the point in $F_i(V)$ with minimum distance to $p_u$. As we have to complete this operation for each device per iteration, we have $n$ query points. In a next step, we show how to evaluate all $n$ queries in time $O(n \log n)$. 
A classical data structure to determine nearest neighbors uses a Voronoi diagram together with a point location algorithm. The Voronoi diagram of a set of points (the sites) is a subdivision of a space into Voronoi cells, one per site with the property that all points inside a Voronoi cell have the same closest site. Subsequently, the Voronoi cell that a query point lies in is determined by a point location algorithm. Unfortunately, the 3-dimensional Voronoi diagram has worst case combinatorial complexity $O(n^2)$. So, computing the whole Voronoi diagram would already yield a running time of $\Theta(n^2)$, and would not be an improvement with respect to the adapted Bellman-Ford algorithm. However, we do not need to compute the Voronoi diagram for the whole 3-dimensional space, since the query point set has a special structure in our setting with all points lying in the plane with $z = 0$. If we only consider this plane, then the Voronoi diagram has a worst case combinatorial complexity of $O(n)$. Therefore, we only compute the Voronoi diagram for the plane with $z = 0$, and refer to it by $\gamma_{i=0}$. 

**Figure 3.6.3:** Encoding of $i$-th iteration for device $u$. 

![Voronoi Diagram](image-url)
3.6. Shortest Paths in Transmission Graphs

To determine $\mathcal{V}_i^{r=0}$, we use the relationship between Voronoi diagrams and the upper envelope of an appropriately defined set of functions (see Chapter 11 in [25] for a thorough description of this relationship). Given a set of functions, the upper envelope is the pointwise maximum of the functions. Let $h_{f_i(v)}(x, y, z)$ be a linear function dependent on the three coordinates of $f_i(v)$, for each device $v \in V$, defined as follows:

$$h_{f_i(v)}(x, y, z) := \left( w = 2f_i(v)_x \cdot x + 2f_i(v)_y \cdot y + 2f_i(v)_z \cdot z \right.$$

$$\left. - \left(f_i(v)^2_x + f_i(v)^2_y + f_i(v)^2_z\right) \right). \quad (3.6.2)$$

The function value $h_{f_i(v)}(x, y, z)$ can be seen as a fourth coordinate, and we refer to it by $w$. Further, let $H_i = \{h_{f_i(v)} : v \in V\}$ be the set of all functions $h_{f_i(v)}$ for all devices $v \in V$, and $\mathcal{U}(x, y, z) := (w = x^2 + y^2 + z^2)$. Each $h_{f_i(v)}$ is a hyperplane in the 4-dimensional space, and it is the tangent hyperplane to $\mathcal{U}$ at the point $(f_i(v)_x, f_i(v)_y, f_i(v)_z, \mathcal{U}(f_i(v)))$, the point "vertically" above the position of $f_i(v)$. The distance between the two functions $\mathcal{U}$ and $h_{f_i(v)}$ at position $p = (x, y, z)$ is

$$\mathcal{U}(p) - h_{f_i(v)}(p) = x^2 + y^2 + z^2 - \left(2f_i(v)_x \cdot x + 2f_i(v)_y \cdot y \right.$$

$$+ 2f_i(v)_z \cdot z - (f_i(v)_x^2 + f_i(v)_y^2 + f_i(v)_z^2) \right)$$

$$= (x - f_i(v)_x)^2 + (y - f_i(v)_y)^2 + (z - f_i(v)_z)^2$$

$$= |p, f_i(v)|^2.$$

The hyperplane $h_{f_i(v)}$ together with $\mathcal{U}$ encodes the distance from $f_i(v)$ to any other point in $\mathbb{R}^3$. Hence, $v$ is the nearest device for point $p = (x, y, z)$ if and only if $h_{f_i(v)}(x, y, z) = \max_{v \in V} h_{f_i(v)}(x, y, z)$.

The upper envelope $E_{H_i}$ contains this information for all possible positions in $\mathbb{R}^3$. It maps each point $p \in \mathbb{R}^3$ to the value of the highest point of intersection between the line orthogonal to $w = 0$ going through $p$ and the functions in $H_i$. If this line does not meet a function, the upper envelope is undefined at this position. The maximization diagram $M_{H_i}$ is the
cell subdivision in the plane with \( w = 0 \), where over each cell the same
single function \( h_{f_i(v)} \) attains the upper envelope. Consequently, the
maximization diagram \( M_{H_i} \) corresponds to the Voronoi diagram \( \mathcal{V}_i \). For our
purposes, we need to compute the Voronoi diagram with sites \( F_i(V) \) for
the plane \( z = 0 \). To do so, we could compute the 3-dimensional Voronoi
diagram via the upper envelope and maximization diagram, and then trans-
form \( \mathcal{V}_i \) to the plane \( z = 0 \). To avoid unnecessary computation, Beier et al.
[11] proved that is is possible to compute the transformation in the \( z = 0 \)
plane first, followed by the computation of the upper envelope and the max-
imization diagram. Algorithm Geometric\( \kappa \)HopsBellmanFord summarizes
the above approach.

**Algorithm 6** Geometric\( \kappa \)HopsBellmanFord: computes \( \kappa \)-hops shortest \( s - t \)-path where devices have unlimited maximal transmission range

1: \( \{M[v, i]\} \) stores lowest cost from source \( s \) to \( v \) using at most \( i \) edges
2: \( M[s, i] = 0 \) and \( M[v, i] = \infty \) for all other \( v \in V \) and all \( i \in [0, \kappa] \)
3: for all \( i = 1, \ldots, \kappa \) do
4: compute set of hyperplanes \( H_i \)
5: compute intersection of each \( h \in H_i \) with hyperplane \( z = 0 \)
6: \( \{H_i^{z=0}\} = \) set of resulting 2-dimensional hyperplanes \( \}
7: \( \mathcal{V}_i^{z=0} = M_{H_i^{z=0}} \)
8: for all \( v \in V \) do
9: point location with \( (v_x, v_y) \) in \( V_i^{z=0} \)
10: \{\( v \) lies in cell of \( \bar{v} \}\}
11: \( M[v, i] = M[\bar{v}, i - 1] + c(\bar{v}, v) \)
12: end for
13: end for
14: output \( M[t, \kappa] \)

**Lemma 3.18** ([11]). The \( \kappa \)-hops shortest \( s - t \)-path problem where all
devices have unlimited maximal transmission range can be solved in time
\( O(\kappa n \log n) \).

**Proof.** We use Algorithm Geometric\( \kappa \)HopsBellmanFord. Consider the
invariant that \( M[v, i] \) stores the lowest cost from source \( s \) to device \( v \) using
at most $i$ edges. The fact that expression 3.6.1 is equivalent to a nearest neighbor problem and the correct evaluation of the nearest neighbor query guarantee that the invariant holds during the algorithm.

The running time of the algorithm depends on the running time of a single iteration for the outer for-loop. So, we consider a fixed iteration. The steps in line 4 and 5 run in time $O(n)$. The maximization diagram $M_{H_\sigma=0}$ represented by a list of all cells with its edges can be computed in time $O(n \log n)$ (see Chapter 11 in [25]). A standard point location algorithm for one device inside the maximization diagram needs time $O(\log n)$, and doing it for all $n$ query points needs time $O(n \log n)$. Hence, a single iteration runs in time $O(n \log n)$. Since the outer loop is executed $\kappa$ times, the total running time adds up to $O(\kappa n \log n)$.

Algorithm Geometric$\kappa$HopsBellmanFord works under the assumption that each device reaches every other device with its maximal transmission range. If we drop this assumption, and use the same dynamic programming approach as before, the evaluation of expression (3.6.1) for a fixed iteration and device reduces to a similar distance problem in the 3-dimensional space but only considering devices that can be reached.

**Observation 3.19.** The problem of computing $M[v, i]$ in expression (3.6.1) reduces to a nearest neighbor problem with sites $f_i(\bar{v})$ and query point $p_v = (v_x, v_y, 0)$ where the sites are restricted to devices $\bar{v}$ with a maximal transmission range large enough to reach device $v$.

So, we need to solve $n$ such nearest neighbor problems per iteration. The goal is to solve them in time almost linear in the number of devices. One possibility is to use the same idea of a Voronoi-like diagram together with a point location algorithm. We subdivide the whole plane with $z = 0$ into cells where each point in a cell has the same nearest neighbor being able to reach the point, and then determine for each of the $n$ query point the cell the point is lying in. The running time of this algorithm is inherently connected to the combinatorial complexity of the subdivision. Unfortunately, the worst case combinatorial complexity of such a subdivision is open. The determination of this combinatorial complexity as well as other
solution methods to derive a near linear running time for one iteration are interesting issues of future research.

3.7 Conclusions and Open Problems

The overall picture of the device placement problem looks as follows. The cases with one additional device to place are solvable in polynomial time independent of the number of commodities. For multiple identical additional devices, the case with multiple commodities is \( \mathcal{NP} \)-hard, whereas the case with a single commodity is polynomial-time solvable. For multiple individual additional devices, the case with two commodities is already difficult, while the case with a single commodity is still open.

As indicated in the introduction, our analysis provides a first step to investigate network creation games in wireless ad hoc networks. In this chapter, we studied the problem from an optimization point of view. As a next step, it would be interesting to study the game, that is, for instance the equilibria, arising when several agents either sequentially or simultaneously enter an existing network, each trying to selfishly optimize its position.

A further major open problem is to provide a fast algorithm for the \( \kappa \)-hops shortest path problem in transmission graphs where devices possess a restricted maximal transmission range. Related to this problem is the combinatorial complexity of the Voronoi-like subdivision arising from the application of the ideas in [11] to the \( \kappa \)-hops shortest path problem in transmission graphs. Lastly, it would be interesting to apply ideas similar to those used in the \( \kappa \)-hops shortest path problem to the pure shortest path problem in transmission graphs.
Chapter 4

Strategic Deletion, Movement, and Range Change of Devices

4.1 Introduction

The placement of new devices is not the only possible operation for a profit-maximizing agent controlling a set of devices in a wireless network with a VCG-like mechanism that uses the shortest path and shortest replacement paths. Many other operations are imaginable. We study some of these operations in this chapter. Generally, we regard all operations with respect to profit maximization (except the delete operation where the profit loss is minimized). As we have seen in Chapter 3.1, profit maximization is equivalent to path cost minimization due to the structure of the payments in the mechanism. To this end, we first consider the device deletion problem in which a set of devices should be deleted from the wireless network without worsening the shortest paths too much. Next, we investi-
gate the problem of moving existing devices in order to improve shortest paths. We refer to this problem as the device movement problem. Subsequently, all device positions remain unchanged, and we consider the range change problem where the maximal transmission ranges of some devices can be adjusted. We close our investigations with looking at another relevant problem in wireless networks. Until now, a source-destination pair is charged a cost for every edge it uses. In contrast, we consider the problem where some commodities can share an edge and pay the cost only once. This problem is called the bundling transmission cost problem.

### 4.1.1 Related Work

Shortest path problems are one of the most prominent problems in computer science. They have been studied since as early as the 1950’s, and still are an active research area. We review the most important results related to shortest path problems in our setting.

The single-source shortest path problem seeks a shortest path from a given source vertex to all other vertices in the graph. For graphs with non-negative edge costs, Dijkstra [28] proposed an algorithm with running time $O(n^2)$. The running time of Dijkstra’s algorithm is inherently connected to the implementation of a priority queue. Using faster implementations of priority queues, the running time of Dijkstra’s algorithm was continually improved, the fastest implementation using Fibonacci heaps as priority queues with running time of $O(m + n \log n)$ [36]. For graphs with arbitrary edge costs, Bellman-Ford’s dynamic program requires time $O(mn)$ [12, 35, 22]. Finding a shortest path for a given pair of vertices has the same running time as the single-source shortest path problem. Another well-known variant is the single-destination shortest path problem where we seek a shortest path to a fixed destination from any vertex. This problem can be reduced to the single-source shortest path problem by reversing the direction of each edge in the graph.

The all-pairs shortest path problem seeks a shortest path from every vertex to every other vertex. A straightforward implementation uses a Dijkstra single source shortest path computation from each vertex resulting
4.1. Introduction

in a running time of $O(n(m + n \log n))$ on a graph with nonnegative edge costs, and a running time of $O(mn^2)$ can be achieved on a graph with arbitrary edge costs from a generalization of Bellman-Ford's algorithm. More sophisticated approaches yield better running times. For sparse graphs with arbitrary edge costs, using the algorithm of Pettie [71] gives a running time $O(mn + n^2 \log \log n)$, and for dense graphs, the algorithm of Takaoka [76] takes time $O(n^3 (\log \log n / \log n)^{1/2})$.

In addition to the classical settings of the single-source shortest path problem and the all-pairs shortest path problem, there exist many variants of these problems such as geometric shortest paths, shortest paths in planar graphs, approximate shortest paths, etc. A variant closely related to our setting is the dynamic shortest paths problem [26]. Here, all-pairs shortest paths are to be maintained in the presence of vertex or edge insertions and removals. Most research in this field considers amortized time analysis, as the main goal is to be prepared for a sequence of such operations. If we are interested in a worst-case analysis, nothing better than a recomputation of the whole distance matrix from scratch was known until recently. In [78], Thorup presented the first algorithm yielding a running time of $\tilde{O}(n^{2.75})$ for a single vertex insertion or removal.

More related work specific to the device deletion problem is listed in the corresponding Section 4.2.

4.1.2 Model and Notation

The model and the notation essentially follows those introduced in Chapter 3.1.2. We are given a transmission graph consisting of a device set and commodities, the number of devices on which an operation can be performed, plus the operation type, and we want to find shortest paths between the commodity set. This chapter studies the following operation types.

Device Deletion  In a device deletion instance, we are given a transmission graph $T = (V, E, K, p, r, c)$, and a positive integer $\Delta n$, and the goal is to delete $\Delta n$ devices such that the profit loss is minimum.
Device Movement In a device movement instance, we are given a transmission graph \( T = (V, E, K, p, r, c) \), and a positive integer \( \Delta n \), and the goal is to find new positions for \( \Delta n \) devices such that the profit is maximum.

Transmission Range Change In a transmission range change instance, we are given a transmission graph \( T = (V, E, K, p, r, c) \), a positive integer \( \Delta n \), and a positive number \( r^+ \), and the goal is to change the maximal transmission range of \( \Delta n \) devices by at most \( r^+ \) each such that the profit is maximum.

Note that there is no differentiation into an individual and an identical problem variant for all operations, since we perform the operations on the existing device set which has diverse maximal transmission ranges.

The model and the notation is extended straightforwardly to the bundling transmission cost problem. In the first instance, we are interested in the problem without any operations. This means that a transmission graph completely describes an instance of the bundling transmission cost problem.

Bundling Transmission Cost In a bundling transmission cost instance, we are given a transmission graph \( T = (V, E, K, p, r, c) \), and the goal is to find a subgraph \( T' = (V', E', K, p, r, c) \) with \( V' \subseteq V \) and \( E' \) based on \( r \) such that each source-destination pair is connected and \( \sum_{e \in E'} c(e) \) is minimum.

### 4.1.3 Summary of Results

We present computational complexity results for many cases of the problem variants. Table 4.1 gives an overview of the results. For completeness, we also listed the results for the device placement problem from Chapter 3. The two parameters in the first column of the table describe the number of devices involved in a operation, and the number of commodities. The first parameter is 1 for the single device case, and \( \Delta n \) for the multiple device
4.1. Introduction

<table>
<thead>
<tr>
<th>$\Delta n, k$</th>
<th><strong>DEVICE PLACEMENT</strong></th>
<th><strong>DEVICE DELETION</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>$1, 1$</td>
<td>$O(n^2)$</td>
<td>$O(n \log n + m)$</td>
</tr>
<tr>
<td></td>
<td>[Section 3.2.2]</td>
<td>[Section 4.2.1]</td>
</tr>
<tr>
<td>$1, k$</td>
<td>$O(k^2 n^{8+\epsilon} \log (kn^{4+\epsilon}))$</td>
<td>$O(k\sqrt{n}(n \log n + m))$</td>
</tr>
<tr>
<td></td>
<td>[section 3.3]</td>
<td>[Section 4.2.2]</td>
</tr>
<tr>
<td>$\Delta n, 1$</td>
<td>identical: $O((\Delta n)^2 n^2)$</td>
<td>$O(nm)$</td>
</tr>
<tr>
<td></td>
<td>[Section 3.2.3]</td>
<td>[Section 4.2.3]</td>
</tr>
<tr>
<td></td>
<td>individual: open</td>
<td></td>
</tr>
<tr>
<td>$\Delta n, k$</td>
<td>identical: $\mathcal{NP}$-hard</td>
<td>$\mathcal{NP}$-hard</td>
</tr>
<tr>
<td></td>
<td>[Section 3.4]</td>
<td>[Section 4.2.4]</td>
</tr>
<tr>
<td></td>
<td>individual: $\mathcal{NP}$-hard already for $k = 2$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[Section 3.5]</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\Delta n, k$</th>
<th><strong>DEVICE MOVEMENT</strong></th>
<th><strong>RANGE CHANGE</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>$1, 1$</td>
<td>$O(n^3)$</td>
<td>$O(n^2)$</td>
</tr>
<tr>
<td></td>
<td>[Section 4.3.1]</td>
<td>[Section 4.4.1]</td>
</tr>
<tr>
<td>$1, k$</td>
<td>$O(k^2 n^{9+\epsilon} \log (kn^{4+\epsilon}))$</td>
<td>$O(n(n \log n + m + k))$</td>
</tr>
<tr>
<td></td>
<td>[Section 4.3.2]</td>
<td>[Section 4.4.2]</td>
</tr>
<tr>
<td>$\Delta n, 1$</td>
<td>open</td>
<td>$O(\Delta n \cdot n^2)$</td>
</tr>
<tr>
<td></td>
<td>[Section 4.4.3]</td>
<td>[Section 4.4.3]</td>
</tr>
<tr>
<td></td>
<td>budget version: $\mathcal{NP}$-hard</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[Section 4.4.5]$^a$</td>
<td></td>
</tr>
<tr>
<td>$\Delta n, k$</td>
<td>$\mathcal{NP}$-hard already for $k = 2$</td>
<td>$\mathcal{NP}$-hard</td>
</tr>
<tr>
<td></td>
<td>[Section 4.3.3]</td>
<td>[Section 4.4.4]</td>
</tr>
</tbody>
</table>

$^a$only interesting for $\Delta n > 1$

**Table 4.1:** Overview over complexity of strategic operations of devices.
case. Similarly, the second parameter is 1 for the single commodity case, and \( k \) for the multiple commodity case. For the bundling transmission range problem, we show that the problem is already difficult without any operations.

The chapter is organized as follows. We explore the different problem variants consecutively. We start with the device deletion problem in Section 4.2, continue with the device movement problem in Section 4.3 and the transmission range change problem in Section 4.4. Within a problem variant, we treat the problem with increasing difficulty. We start with the case of one single device and a single commodity, then consider the case with a single device and multiple commodities. Next, we investigate the case with multiple devices and one commodity. The most difficult case with multiple devices and multiple commodities is explored as a last case. Finally, Section 4.5 investigates the bundling transmission range problem.

### 4.2 Device Deletion

The device deletion problem asks for deleting a fixed number of devices such that the increase with respect to the total cost of shortest paths is minimum. More precisely, an instance of the device deletion problem consists of a transmission graph \( T = (V, E, K, p, r, c) \), and a positive integer \( \Delta n \). The goal is to delete \( \Delta n \) devices such that \( SP(T') \) is minimized, where \( T' \) is the transmission graph without the deleted devices.

#### Related Work

So far, the related problem of finding the most vital node of a shortest path has been studied quite intensively. The most vital node of a shortest path between a pair of vertices is a vertex whose deletion incurs the largest increase in path cost for this path. Nardelli et al. [63] considered the most vital node problem in an undirected graph and presented an algorithm with running time \( O(n \log n + m) \). The more general problem where we are to find the \( k' \) most vital nodes for a given \( k' \) was proved to be \( \mathcal{NP} \)-hard in
4.2. Device Deletion

[10]. The version with removing edges instead of vertices on the shortest path exists as well, and has the same time complexity.

A further related problem is the replacement path problem. In the replacement path problem, we are given a graph with non-negative edge costs, two distinct vertices $s$, $t$ together with the shortest path between them. The goal is to compute the shortest $s - t$-path in each of the graphs resulting from removing one of the edges on the shortest path. For undirected graphs, the algorithm of Nardelli et al. [63] to solve the most vital node problem solves the replacement path problem as well. Hershberger and Suri [45, 46] proposed an alternative algorithm for the replacement path problem with the same running time. For directed graphs, the best known algorithm for the replacement path problem is the naïve algorithm with running time $O(n(n \log n + m))$, stemming from executing a single source shortest path algorithm up to $n$ times.

Demetrescu et al. [27] studied a more general problem. They consider the problem of answering shortest path queries from any vertex to any other vertex avoiding an arbitrary vertex in between, and proposed two approaches for the problem. The better approach with respect to running time needs preprocessing time of $O(n^{2.5} \log n + n^{1.5}m)$, and subsequently answers any query in constant time.

4.2.1 Single Device Deletion for Single Commodity

A straightforward solution for the case of a single device and a single commodity is to execute the following procedure for every device in the transmission graph: temporarily remove the device and compute the shortest $s - t$-path from scratch. We then remove the device achieving the minimum shortest path. In other words, we remove device $x$ for which $x = \arg \min_{1 \leq v \leq n} SP_{T-v}(s, t)$, where $T-v$ is the transmission graph with device $v$ and all edges adjacent to $v$ removed.

**Observation 4.1.** The single device deletion problem for a single commodity can be solved in time $O(n(n \log n + m))$.

**Proof.** Correctness follows from the fact that we check all possible cases.
The approach described above consists of $n$ separate shortest path computations. Using Dijkstra’s shortest path algorithm implemented with Fibonacci heaps [36] for each of the $n$ device removals leads to a running time of $O(n \log n + m)$.

Observe that for the cases where there is at least one device not used in the shortest $s-t$-path, we can immediately delete one of these devices to solve the problem. We can use this observation for a faster algorithm. If not all devices are part of a shortest path, then we delete one of the devices not used. If all devices are part of the shortest path, then we exploit the particular structure of the shortest path.

**Theorem 4.2.** The single device deletion problem for a single commodity can be solved in time $O(n \log n + m)$.

**Proof.** We compute the single source shortest path tree from $s$, and the single destination shortest path tree to $t$. We distinguish two cases depending on the structure of the computed shortest $s-t$-path. If there is a device not used in the shortest path then we delete this device. Otherwise, we know that the shortest path visits all $n$ devices. Suppose that the shortest path is $SP(s,t) = \{s = \sigma(1), \sigma(2), \ldots, \sigma(n-1), \sigma(n) = t\}$. When a device $\sigma(i)$, for $1 < i < n$, and its incident edges are deleted, the shortest path in the resulting graph has a simple structure. The shortest path consists of a source path $\{s, \ldots, \sigma(i')\}$ going from $s$ to $\sigma(i')$ with $i' < i$, a detour edge from $(\sigma(i'), \sigma(i''))$ circumventing $\sigma(i' + 1), \ldots, \sigma(i), \ldots, \sigma(i'' - 1)$, and a destination path $\{\sigma(i''), \ldots, t\}$ going from $\sigma(i'')$ to $t$ with $i < i''$. Note that fixing the detour edge determines the source and destination path. Let $E_D$ be the set of detour edges. We then delete a device which is not on the path for which

$$
\min_{(\sigma(i'), \sigma(i'')) \in E_D} \left( \sum_{j=1}^{i'-1} c(\sigma(j), \sigma(j+1)) + c(\sigma(i'), \sigma(i'')) \right) + \sum_{j=i''}^{n-1} c(\sigma(j), \sigma(j+1)) \tag{4.2.1}
$$
4.2. Device Deletion

is achieved.

The algorithm works correctly since we check all possible detour edges. The time complexity follows from the time needed for the single-source shortest paths computation and the time needed to evaluate (4.2.1) for all detour edges. The single-source shortest paths can be computed in time $O(n \log n + m)$. We have to check for each edge whether it is a detour edge, and if yes, evaluate expression (4.2.1). This can be done in constant time for each edge using the values from the path tree computation. In total, the running time adds up to $O(n \log n + m)$.

4.2.2 Single Device Deletion for Multiple Commodities

For the case in which we have multiple commodities and one single device to delete, the solution method from Section 4.2.1 still works: remove a device $x$ for which $x = \arg \min_{1 \leq v \leq n} \sum_{i=1}^{k} SP_{T-v}(s_i, t_i)$, where $T-v$ is the transmission graph with device $v$ and all edges adjacent to $v$ removed. This approach runs in time $O(n \cdot \tau_{SP(K)})$ where $\tau_{SP(K)}$ is the time needed to compute the shortest paths for all commodities in $K$.

In essence, the straightforward algorithm implicitly computes a two-dimensional table with $k$ rows and $n$ columns in which it stores, for each commodity and each device, the total cost of a shortest path for this commodity avoiding that device. Finally, it accumulates the entries of each column, that is, of each device, and deletes the device achieving the minimum value. An algorithm from [27] allows to compute this table faster. The algorithm in [27] provides seven data structures from which a single entry of such a table can be computed in constant time. In fact, this algorithm allows the computation for all device pairs, that is, the case where each device pair forms a commodity would be the equivalent case in our setting. The running time of the algorithm in [27] is dominated by the construction time of a single one of these data structures, and amounts to $O(n^{1.5} (n \log n + m))$. Roughly speaking, the running time comes from the fact that there are $\sqrt{n}$ shortest path tree computations for each device. In our setting, we only need to know the distance table avoiding a device
for all source-destination pairs instead of for all device pairs. A closer inspection of [27] shows that the same data structure is also the dominating factor in this setting, yielding the following theorem.

**Theorem 4.3.** The single device deletion problem for multiple commodities can be solved in time $O(k\sqrt{n}(n \log n + m))$.

**Proof.** Correctness follows from the proof in [27].

The construction of the table needs time $O(k\sqrt{n}(n \log n + m))$. The accumulation of all entries for a device needs time $O(k)$, yielding time $O(nk)$ for all devices. 

---

### 4.2.3 Multiple Device Deletion for Single Commodity

In this case, an optimal solution is equivalent to a shortest path using at most $n - \Delta n$ devices. We can then delete $\Delta n$ devices not used in such a path. To find a shortest path over at most $n - \Delta n$ devices, the dynamic programming approach in Algorithm 5 from Section 3.6.2 can be used.

**Theorem 4.4.** The multiple device deletion problem for a single commodity can be solved in time $O(nmn)$.

**Proof.** We use the dynamic programming approach in Algorithm 5 from Section 3.6.2 with $\kappa$, the number of maximal allowed hops, set to $n - \Delta n - 1$.

---

### 4.2.4 Multiple Device Deletion for Multiple Commodities

Next, we prove that the general case with multiple devices and multiple commodities is $\mathcal{NP}$-hard. The reduction to prove $\mathcal{NP}$-hardness of this problem uses the same ideas as the reduction to prove $\mathcal{NP}$-hardness of IDENTICAL DEVICE PLACEMENT from Section 3.4.
4.2. Device Deletion

**Problem:** Device Deletion

**Instance:** An instance $I = (T', \Delta n, Z)$ of Device Deletion consists of a transmission graph $T = (V, E, K, p, r, c)$, a positive integer $\Delta n$, and a positive number $Z$.

**Question:** Is there a deletion of $\Delta n$ devices such that the difference $SP(T') - SP(T)$ is at most $Z$, where $T'$ is the transmission graph after the deletion of the $\Delta n$ devices?

**Theorem 4.5.** Device Deletion is $\mathcal{NP}$-hard.

**Proof.** The proof is via a reduction from planar Exact Cover By 3-Sets where each element appears in either two or three sets (planar X3C-3). Let $I(U, S, b)$ be an instance of planar X3C-3 with $U$ a set of $3b$ elements, $S = \{S_1, \ldots, S_{|S|}\}$ a collection of 3-element subsets of $U$, and budget $b$. We ask whether there is a subcollection of size $b$ from $S$ covering $U$. Planar X3C-3 is known to be $\mathcal{NP}$-complete [30].

The main construction is essentially the same as in the hardness proof for the identical device placement problem. We only sketch the important parts of the proof. For the details, such as the exact positioning of the devices, we refer to the $\mathcal{NP}$-hardness proof of the identical device placement problem in Section 3.4.

We start with the description of the device and edge set of the transmission graph. Figure 4.2.1 shows a schematic illustration of an instance. In detail, all devices are embedded in the plane in exactly the same way as in the $\mathcal{NP}$-hardness proof of the identical device placement problem. The device set $V$ consists of element devices, set devices, the global destination device, element-set chain devices, and set-destination chain devices. The set $V^e$ of element devices contains one device $v^e_i$ per element $i \in U$, and the set $V^s$ of set devices one device $v^s_j$ per set $S_j \in S$. The global destination device $v^T$ is a single device. In Figure 4.2.1, the black dots in the top row are the element devices, the black dots in the third row from the top are the set devices, and the single black dot in the bottom is the global destination device. We construct an element-set chain from an element device $v^e_i$
Figure 4.2.1: Schematic overview illustration of the reduction for the device deletion problem.
to a set device $v_j^g$ if $i$ is member of $S_j$. Such an element-set chain consists of multiple devices placed very close together one after the other. We denote the set of devices on the element-set chain between an element device $v_i^e$ and a set device $v_j^g$ by $V_{i,j}$. Due to the particular choice of the positions and the maximal transmission ranges of these devices, it is only possible to communicate in one direction through the chain. Although its exact structure is essential for the construction to work, a chain can be seen as one large device in the transmission graph. The circles in the second row from the top in Figure 4.2.1 correspond to these chains. Similarly, we construct a set-destination chain from each set device $v_j^g$ to the global destination device, for $j \in \{1, \ldots, |S|\}$. These set-destination chains are again built by multiple appropriately placed devices. We denote the set of devices on the set-destination chain from $v_j^g$ to the global destination device by $V_{j,T}$. In the figure, the set-destination chains correspond to the circles in the fourth row from the top. The directed edges in the transmission graph indicate the possible communication directions.

Up to here the construction is exactly the same as the construction in
the hardness proof for the identical device placement problem. We next explain the differences. In contrast to the device placement proof, there are no empty large gaps on the set-destination chains. Instead, we already place a device in the middle of each large gap. An extract with a single set-destination chain for a set is shown in Figure 4.2.2. There are five distinct devices: the set device $v_j^S$, the device $v_j^{UG}$ above the gap, the device $v_j^{gap}$ in the gap, the device $v_j^{LG}$ below the gap, and the global destination device $v^T$. The set device $v_j^n$ is connected to $v_j^{UG}$ via multiple devices, $v_j^{gap}$ is exactly in the middle between $v_j^{UG}$ and $v_j^{LG}$, and the device $v_j^{LG}$ is connected to $v^T$ via multiple devices.

Before we proceed with the commodity set, we state some facts about the transmission graph that stem from the exact embedding of the various devices. No two element-set chains and no two set-destination chains cross each other. Crossings between an element-set chain and a set-destination chain exist, but the maximal transmission ranges are chosen in such a way that communication is only possible from a device on the set-destination chain to a device on the element-set chain, and not in the other direction. There is a path of cost 2 along the element-set chain from each element device to the set devices the element is a member of. From each set device, there is a path of cost $W$ along the set-destination chain to the global destination device.\footnote{The exact value of $W$ is computed in Section 3.4. For our purposes here, it suffices that $W$ is bounded by a polynomial in the input parameters, and that it is equal for all sets.} By construction, there exists a path with equal cost between each element device and the global destination device, namely a path of cost $2 + W$. This cost is attained on a path from the element device via a set device the element is a member of to the global destination device. Note that there is more than one such path per element, namely either two or three, depending on the number of occurrences of an element. Further, there exist other paths that connect an element device with the global destination device. All these alternative paths have the same structure, going from an element device to a set device in which the element occurs. Next, the path goes from this set device via a crossing between a set-destination chain and an element-set chain to another set device, and finally goes possibly through some other set devices to the global destination device. The
4.2. Device Deletion

The cost of such a path is strictly larger than that of a direct path via a single set device. Any other path, for instance a path which partially goes from a set device to an element device, is prohibited through a careful choice of the maximal transmission ranges of the devices between any element and set device.

As a next ingredient, we introduce the commodities. The set of commodities is more involved than in the construction for the device placement problem. The commodity set consists of two types of commodities, the element commodities $K^e$ and the auxiliary commodities $K^a$. The set $K^e$ of element commodities is the same as in the hardness proof of the identical device placement problem. There are as many element commodities as elements, each element device is the source of a commodity, and all commodities share the global destination device $v^T$. The auxiliary commodities $K^a$ take care that only devices $v^{gap}_j$, for $1 \leq j \leq |S|$, are device candidates to be deleted. More precisely, the auxiliary commodity set $K^a$ consists of the following two types of commodities.

**Element-set commodities** If element $i \in S_j$, then $K^a$ contains a commodity with the element device $v^e_i$ as source, and the set device $v^s_j$ as destination. This results in $3 \cdot |S|$ additional commodities.

**Set-destination commodities** Between each set device $v^s_j$, $1 \leq j \leq |S|$, and the global destination device $v^T$, we add another two commodities. We introduce the commodities $(v^s_j, v^U_j^G)$ and $(v^I_j^G, v^T)$. This contributes another $2 \cdot |S|$ commodities to the commodity set $K^a$.

**Observation 4.6.** The deletion of a device in $v \in V^{i,j}$, for $1 \leq i \leq |U|$ and $1 \leq j \leq |S|$, incorporates shortest path cost strictly larger than two for an element-set commodity in $K^a$.

**Proof.** Consider an element-set commodity in $K^a$ from $v^e_i$ to $v^s_j$. If we delete a device on this element-set chain, then the path along devices $V^{i,j}$ is not possible anymore due to the chosen maximal transmission ranges. If
Chapter 4. Strategic Deletion, Movement, and Range Change

there still exists a path for this commodity, then it goes along an element-set chain $V^{i,j'}$ to a set device $v^{s}_{j'}$, with $j' \neq j$. The path continues from the set device $v^{s}_{j'}$ along a set-destination chain, switches at a crossing to an element-set chain, and reaches another set device. It repeats the pattern “set-destination chain, switch to element-set chain, element-set chain” until it reaches $v^{s}_{j}$. The total cost of such a path is strictly larger than two.

\[ \square \]

**Observation 4.7.** The deletion of a device in $v \in V^{j,T}$, for $1 \leq j \leq |S|$, incorporates infinite shortest path cost for a set-destination commodity in $K^{a}$.

**Proof.** Consider a set-destination commodity in $K^{a}$ from $v^{s}_{j}$ to $v^{UG}_{j}$. Note that any path between the set device and the global destination device contains the path along devices in $V^{j,T}$ as a partial path. Any path without this partial path is prohibited as it is not possible to switch from an element-set chain to a set-destination chain. Hence, the deletion of a device in $V^{j,T}$ between $v^{s}_{j}$ and $v^{UG}_{j}$ disconnects the source and the destination of the commodity, yielding cost of infinity for this commodity. Next, consider a set-destination commodity in $K^{a}$ from $v^{LG}_{j}$ to $v^{T}$. Due to the chosen maximal transmission range, there is only one possible path in the original transmission graph. Thus, deletion of a device in $V^{j,T}$ between $v^{LG}$ and $v^{T}$ disconnects the source and the destination of the commodity, yielding again cost of infinity for this commodity.

\[ \square \]

The sum of the shortest path costs over all commodities in $K^{a}$ adds up to $6|S| + |S| \cdot W'$, where $W'$ is the total cost of a shortest path between a set device and the global destination device minus the cost of a shortest path between $v^{UG}_{j}$ and $v^{LG}_{j}$ for some $j \in \{1, \ldots, |S|\}$.

Finally, we set $\Delta n$, the number of devices to delete, to $|S| - b$, and $Z$ to zero. This completes the construction of the device deletion instance. The whole construction is computable in polynomial time.

We claim that there is a solution for the X3C-3 instance if and only if there is solution for the device deletion instance. We first show the forward direction, that is, given a solution for the X3C-3 instance, we can construct
4.2. Device Deletion

a solution for the device deletion problem. Let the indices of the \( b \) sets in the X3C-3 solution be \( \text{sol} = (\text{sol}_1, \ldots, \text{sol}_b) \). Consider the removal of the devices \( v_i^{\text{gap}}, \) for \( i \in S \setminus \{\text{sol}_1, \ldots, \text{sol}_b\} \). For each element commodity there is still a shortest path of the same cost as before the removal, namely from the element device to the set device the element is a member of, and which is in \( \text{sol} \). Thus, the shortest path cost over all commodities has not changed. Next, we show the reverse direction by showing that if there is no cover of size \( b \), then there is no solution for the device deletion problem. Suppose no cover with size \( b \) exists, and consider any device deletion in the constructed instance. If a device in \( V \setminus (\bigcup_{j=1}^{\left|S\right|} v_j^{\text{gap}}) \) is deleted, then the cost for a commodity is larger than in the original transmission graph due to Observations 4.6 and 4.7. If a device \( v_j^{\text{gap}}, \) for \( 1 \leq j \leq |S| \) is deleted, then there exists an element commodity for which all (two or three) shortest paths in \( T \) are no longer possible in the transmission graph without the deleted devices. The shortest path for this commodity now goes over some other set devices, and its cost will in any case be higher than before. Hence, the total path costs over all commodities has increased, and the instance has no solution. \( \square \)

4.2.5 Device Deletion in a Subset

In a variant of the device deletion problem, we only control a subset \( \tilde{V} \) of the devices, and we are allowed to delete devices from this subset. The algorithms for a single device where we iteratively compute the increase with respect to shortest path cost of a removal of a device in \( \tilde{V} \) are still applicable in this variant. The running time changes to \( O(\tilde{V}(n \log n + m)) \) for the single commodity case and to \( O(\tilde{V} \cdot \tau_{\text{SP}(K)}) \) for the multiple commodity case, as we only iterate over devices under control. The algorithm from Theorem 4.2 for a single device and a single commodity fails, as the special structure of the shortest path does not exist anymore. It is not clear whether it is still possible to solve the problem with a single device and single commodity in time \( O(n \log n + m) \).

The results for the cases with multiple devices to delete still hold with
minor changes. The Bellman-Ford like dynamic programming algorithm for the case with multiple devices and a single commodity needs to take into account the number of used controlled devices. Let $M[v, i, j]$ denote the $i$-hops shortest path from the source $s$ to $v$ using $j$ devices from the controlled device set $\bar{V}$. The recursive formula for $M[v, i, j]$ now depends on whether a device is from $\bar{V}$ or not:

$$
M[v, i, j] = \begin{cases}
  v \in \bar{V} & : \min \left\{ M[v, i - 1, j], \min_{(\bar{v}, v) \in E} \left( M[\bar{v}, i - 1, j - 1] + c(\bar{v}, v) \right) \right\} \\
  v \in V \setminus \bar{V} & : \min \left\{ M[v, i - 1, j], \min_{(\bar{v}, v) \in E} \left( M[\bar{v}, i - 1, j] + c(\bar{v}, v) \right) \right\}
\end{cases}
$$

To incorporate the new recursive formula, the corresponding algorithm is extended by an additional for-loop iterating over $j$, and the running time increases by a factor $|\bar{V}|$ to $O(|\bar{V}|nm)$. The $\mathcal{NP}$-hardness proof for the case with multiple devices and multiple commodities becomes less involved, as we do not need the auxiliary commodities anymore. The subset of controlled devices in the reduction is simply restricted to the gap devices.

### 4.3 Device Movement

In this section, we study the device movement problem. Here, we are allowed to move devices in order to improve shortest paths. The formal setting is as follows: Given a transmission graph $T = (V, E, K, p, r, c)$, and a positive integer $\Delta n$, the goal is to assign new positions to $\Delta n$ devices such that $SP(T')$ is minimized, where $T'$ is the transmission graph with the $\Delta n$ devices moved to their new positions.

#### 4.3.1 Single Device Movement for Single Commodity

Similar to the same case in the device deletion problem, we can iteratively run the following procedure for each device: Temporarily remove the de-
vice, place it optimally, and compute the shortest path in this setting. We then move the device achieving the shortest path with minimum cost.

**Theorem 4.8.** The single device movement problem for a single commodity can be solved in time $O(n \cdot \tau_{DPP}(1,1))$, where $\tau_{DPP}(1,1)$ is the time to optimally place an additional device for one commodity.

**Proof.** The algorithm iterates over all possible cases, and thus is correct.

The algorithm consists of $n$ iterations. In each single iteration, we need $\tau_{DPP}(1,1)$ to compute the best position for the chosen device. □

Using the $O(n^2)$-approach from Section 3.5 to place a single device for a single commodity, we obtain an algorithm with running time $O(n^2)$

### 4.3.2 Single Device Movement for Multiple Commodities

When we are to move a single device for multiple commodities, the above approach still works.

**Theorem 4.9.** The single device movement problem for multiple commodities can be solved in time $O(n \cdot \tau_{DPP}(1,k))$, where $\tau_{DPP}(1,k)$ is the time to optimally place an additional device for multiple commodities.

**Proof.** Correctness of the algorithm follows since we investigate all possibilities.

As there are $n$ iterations of the single device placement for multiple commodities, the stated running time results. □

Since the optimal placement of a single device with multiple commodities can be accomplished in time $O(k^2 n^{8+\varepsilon} \log (kn^{4+\varepsilon}))$ with the algorithm from Section 3.3, the single device movement for multiple commodities is solvable in time $O(k^2 n^{3+\varepsilon} \log (kn^{4+\varepsilon}))$. 
4.3.3 Multiple Device Movement

The status of the multiple device movement problem for a single commodity is still open. Next, we determine the complexity of the multiple device movement problem for two commodities. The decision version of this case is as follows.

**Problem:** DEVICE MOVEMENT FOR TWO COMMODITIES

**Instance:** An instance $I = (T, \Delta n, Z)$ of DEVICE MOVEMENT FOR TWO COMMODITIES consists of a transmission graph $T = (V, E, K, p, r, c)$ with $|K| = 2$, a positive integer $\Delta n$, and a positive number $Z$.

**Question:** Is there a movement of $\Delta n$ devices, such that $SP(T') = SP_{T'}(s_1, t_1) + SP_{T'}(s_2, t_2)$ is at most $Z$, where $T'$ is the transmission graph after moving the devices?

Using the same ideas as in the $NP$-hardness proof of the individual device placement problem for two commodities in Section 3.5, we can state the next theorem.

**Theorem 4.10.** DEVICE MOVEMENT FOR TWO COMMODITIES is $NP$-hard.

**Proof.** The proof uses almost the same reduction from PARTITION as the $NP$-hardness proof of the individual device placement problem for two commodities in Section 3.5. We are given a PARTITION instance with a set $A = \{a_1, \ldots, a_{|A|}\}$ of positive integer numbers adding up to $B = \sum_{a_i \in A} a_i$, and we have to decide whether there is a subset $A' \subseteq A$ with $\sum_{a_i \in A'} = B/2$. We construct an instance of DEVICE MOVEMENT FOR TWO COMMODITIES with $4 + |A|$ devices. Device 1 is at position $\langle 0, 0 \rangle$, device 2 at position $\langle B/2 + 1, 0 \rangle$, device 3 at position $\langle 0, M \rangle$, and device 4 at position $\langle B/2 + 1, M \rangle$, with $M$ an integer much larger than $B$. The maximal transmission ranges of these devices are set to one. The remaining $|A|$ devices are all at the same position $\langle 5M, 5M \rangle$, and their maximal transmission range $r(u)$ is set to $a_{u-4}$, for $u = 5, \ldots, 4 + |A|$. 
4.4. Transmission Range Change

Further, device pairs \((1, 2)\) and \((3, 4)\) form each one of the two commodities. The number of moveable devices is set to \(|A|\), and \(Z = 2 + \sum_{a_i \in A} a_i^2\). A solution of the partition problem directly gives a solution for the device movement problem for two commodities. We place the devices with index in \(A'\) one after another on the line segment between \(s_1\) and \(t_1\), and the remaining similarly between \(s_2\) and \(t_2\). If there is no solution for the partition problem, then at least one of the source-destination pair is not connected yielding a total path cost of infinity.

The overall picture for the device movement problem resembles the one for the individual device placement problem. The cases with one moveable device are solvable in polynomial time. The status of the problem with multiple moveable devices and a single commodity is unclear, and the problem with multiple moveable devices already becomes hard for two commodities.

4.3.4 Device Movement in a Subset

The variant where an additional input set \(\tilde{V}\) determines the devices which are allowed to be moved does not differ much from the problem without this parameter. In the single device cases, the iterative algorithms still work. However, we only need to iterate over the controlled devices, and the running times change to \(O(|\tilde{V}| \cdot \tau_{DPP(1,1)})\) for a single commodity respectively to \(O(|\tilde{V}| \cdot \tau_{DPP(1,k)})\) for multiple commodities. In the multiple devices cases, the status of the single commodity case is open as well, and the hardness of the device movement problem for multiple commodities carries over to this variant.

4.4 Transmission Range Change

This section deals with the problem variant where all device positions are fixed, and we are allowed to change the maximal transmission range of some devices. The range change problem is formally defined as follows:
Given a transmission graph \( T = (V, E, K, p, r, c) \), a positive integer \( \Delta n \), and a positive rational \( r^+ \), the goal is to minimize \( SP(T') \), with \( T' \) the transmission graph where the maximal transmission range is increased by \( r^+ \) for each of \( \Delta n \) devices.

### 4.4.1 Single Device Change for Single Commodity

We first investigate the case where we are allowed to change one maximal transmission range with a single commodity. The straightforward iteration over all devices works in this case: We temporarily increase the maximal transmission range of the device by \( r^+ \), and compute the shortest path in this setting. Finally, we select the device achieving the path with minimum cost. This algorithm runs in time \( O(n(n \log n + m)) \). An alternative algorithm computes the desired result faster, as the following theorem shows.

**Theorem 4.11.** The single device range change problem for a single commodity can be solved in time \( O(n^2) \).

**Proof.** We construct an expanded 2-layer graph as follows. Each of the two layers contains a copy of the original transmission graph. A vertex \((u, 0)\) on layer 0 corresponds to device \(u\) and no device with a changed maximal transmission range used on the subpath from source \(s\) to \(u\), whereas a vertex \((u, 1)\) on layer 1 corresponds to the same device \(u\) and one device with a changed maximal transmission range used on the subpath from source \(s\) to \(u\). Edges between the layers are directed from layer 0 to layer 1, and such an edge exists if device \(u\) can reach device \(v\) with its increased maximal transmission range. That is, vertex \((u, 0)\) is connected to vertex \((v, 1)\) if \(r(u) < |uv| \leq r(u) + r^+\), for \(u, v \in V\). The cost of an edge from \((u, 0)\) to \((v, 1)\) is equal to \(|uv|^2\), the transmission cost between \(u\) and \(v\). We now search a shortest path from \((s, 0)\) to \((t, 0)\) and one from \((s, 0)\) to \((t, 1)\), and select the shorter of the two. By construction any path from \((s, 0)\) to \((t, h)\), for \(h = \{0, 1\}\), uses at most one increased maximal transmission range. Hence, the solution is correct.

The construction of the graph needs time \(O(n^2)\). A shortest path can be found in time \(O(n' \log n' + m')\) where \(n' = 2n\) and \(m'\) is the number
of edges in the 2-layer graph. The time to construct the graph dominates the shortest path search regardless of the exact value for $m'$. 

### 4.4.2 Single Device Change for Multiple Commodities

Next, we consider the problem where we are allowed to change the maximal transmission range of a single device for multiple commodities. A straightforward approach where we iteratively change the maximal transmission range of one device and compute the effect yields a running time of $O(n \cdot \tau_{SP(K)})$, where $\tau_{SP(K)}$ is the time needed to compute the shortest path costs for all commodities in $K$. The execution of a single iteration can be accelerated because the shortest path after changing the maximal transmission range of one device has a special structure. More precisely, after changing the maximal transmission range of one device, the shortest path is either equal to the shortest path in the original transmission graph or goes through the modified device. We exploit this structure in the next theorem.

**Theorem 4.12.** The single device range change problem for multiple commodities can be solved in time $O(n(n \log n + m + k))$.

**Proof.** We again iteratively change the maximal transmission range of a single device, and compute the sum of shortest path costs over all commodities. Consider a single iteration, and change the maximal transmission range of device $v$. This change induces a shorter path for a commodity if the induced path uses an outgoing edge of $v$ which does not exist in the original transmission graph. Hence, we run a shortest path tree computation from device $v$, and a shortest path tree computation with $v$ as destination in the modified graph. As a result, we obtain a shortest path from every device to every other device going through $v$. We then compare for each commodity $(s_i, t_i)$ the shortest $s_i - t_i$-path in the original transmission graph with the shortest $s_i - t_i$-path going through $v$ in the modified transmission graph, and select the shorter of the two. Finally, we sum up the cost of the selected paths over all commodities.
Chapter 4. Strategic Deletion, Movement, and Range Change

The algorithm contains a shortest path computation for each commodity in the original transmission graph and the evaluation of each of the \( n \) settings. The running time for the former step is dominated by the one for the latter step. The running time of the latter step is as follows. In a single iteration, we first compute two shortest path trees yielding a running time of \( O(n \log n + m) \). Next, we compare the effect for each commodity. Since the comparison needs constant time, we need time \( O(k) \) for all commodities. Overall, we obtain a running time of \( O(n(m + n \log n + k)) \).

4.4.3 Multiple Device Change for Single Commodity

When we are to change the maximal transmission range of multiple devices for a single commodity, we can use the expanded layer graph approach from Section 4.4.1.

Theorem 4.13. The multiple device range change problem for a single commodity can be solved in time \( O(\Delta n \cdot n^2) \).

Proof. We use an expanded \((\Delta n + 1)\)-layer graph in a similar way as in Lemma 4.11 for the single commodity case. A layer encodes the number of devices for which the maximal transmission range has been changed. Each layer contains a copy of the original transmission graph. A vertex \((u, h)\) on layer \( h \) now corresponds to device \( u \) and exactly \( h \) devices with increased maximal transmission range visited on the subpath from \( s \) to \( u \). There are directed edges from layer \( h \) to the next upper layer \( h + 1 \), for \( 0 \leq h < \Delta n + 1 \). The directed edge from vertex \((u, h)\) to vertex \((v, h + 1)\) exists if \( u \) does not reach \( v \) with its original maximal transmission range, but reaches it with a maximal transmission range increased by \( r^+ \). More formally, \(((u, h), (v, h + 1))\) is a directed edge in the expanded graph if \( r(u) < |uv| \leq r(u) + r^+ \), for \( u, v \in V \), and \( 0 \leq h < \Delta n + 1 \). The cost of such an edge is equal to the transmission cost \(|uv|^2 \) between devices \( u \) and \( v \). We ask for a shortest path in this expanded graph between \((s, 0)\) and \((t, h)\), for any \( h \in \{0, \ldots, \Delta n\} \).
4.4. Transmission Range Change

The construction guarantees that any path between the source and the destination uses at most $\Delta n$ changed maximal transmission ranges. Note that a path may contain the same vertex more than once on different layers. However, the total cost of such a path is strictly larger than the total cost of the path excluding the partial path between the two visits of the same vertex.

The running time follows from the time needed to construct the graph and the time needed to find a shortest path. We need $O(\Delta n \cdot n^2)$ for the construction and $O(n' \log n' + m')$, with $n' = \Delta n \cdot n$ and $m'$ equal to the number of edges in the multi-layer graph, for the computation of a single source shortest path tree from $(s, 0)$. The time for the construction of the multi-layer graph dominates the time to find the shortest path tree regardless of the exact value of $m'$.

\[ \square \]

4.4.4 Multiple Device Change for Multiple Commodities

Next, we determine the complexity of the general range change problem with multiple devices and multiple commodities. The decision version of this case is as follows.

PROBLEM: RANGE CHANGE

INSTANCE: An instance $I = (T, \Delta n, r^+, Z)$ of RANGE CHANGE consists of a transmission graph $T = (V, E, K, p, r, c)$, a positive integer $\Delta n$, and positive numbers $r^+$ and $Z$.

QUESTION: Is there a transmission range change by $r^+$ for each of $\Delta n$ devices such that the difference $SP(T) - SP(T')$ is at least $Z$, where $T' = (V, E, K, p, r', c)$ is the transmission graph with the new assigned maximal transmission ranges?

Theorem 4.14. RANGE CHANGE is $\mathcal{NP}$-hard.

Proof. We use the same reduction as we used in Section 3.4 to prove $\mathcal{NP}$-hardness of the identical device placement problem, but with devices
already placed in the middle of each large gap on the big chains. The maximal transmission range of these devices is set to zero. Further, the parameter \( r^+ \) is set to \( d_{gap}/2 \). The proof then goes along the same lines as the hardness proof of the identical device placement problem.

\[ \square \]

### 4.4.5 Multiple Device Change for Single Commodity with Budget

In a variation of the range change problem, we bound the total increase of the maximal transmission ranges by a budget \( R \), instead of the increase for a single device together with the number of changeable devices. The decision version of this variation is defined as follows.

**Problem:** Range Change for Single Commodity with Budget

**Instance:** A transmission graph instance \( T = (V, E, K, p, r, c) \) with \( |K| = 1 \), and positive numbers \( R, Z \).

**Question:** Is there an assignment \( r' \) of maximal transmission ranges such that \( \sum_{v \in V} (r'(v) - r(v)) \leq R \), and \( SP(T') = SP_{T'}(s, t) \) is at most \( Z \), where \( T' = (V, E, K, p, r', c) \) is the transmission graph with the new assigned maximal transmission ranges \( r' \)?

**Theorem 4.15.** Range Change for Single Commodity with Budget is \( \mathcal{NP} \)-hard.

**Proof.** We use a reduction from the subset sum problem, which is known to be weakly \( \mathcal{NP} \)-complete (see SP13 in [38]). In the subset sum problem, we are given a set \( A = \{a_1, \ldots, a_{|A|}\} \) of positive integer numbers, and a positive integer \( B \). We have to decide whether there is a subset \( A' \subseteq A \) such that \( \sum_{a_i \in A'} a_i = B \). We construct an instance of Range Change for Single Commodity with Budget as follows. For each \( a_i \), we build a gadget \( i \) consisting of seven devices \( v_{i,j} \), for \( i = 1, \ldots, |A|, j = 0, \ldots, 6 \). We first describe the device positions in relative coordinates, and later specify the effective positions (see Figure 4.4.1). Let \( a^+ = \max_i a_i \).
4.4. Transmission Range Change

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure.png}
\caption{A gadget of the reduction for one \(a_i\).}
\end{figure}

- Device \(v_{i,0}\) is at position \(\langle 1, 0 \rangle\) and has a maximal transmission range of zero.
- Device \(v_{i,1}\) is at position \(\langle 0, 0 \rangle\) and has a maximal transmission range of one.
- Device \(v_{i,2}\) is at position \(\langle 0, 1 \rangle\) and has a maximal transmission range of \(\sqrt{a_i/2 - 1}\).
- Device \(v_{i,3}\) is at position \(\langle 0, \sqrt{a_i/2 - 1} + 1 \rangle\) and has a maximal transmission range of \(a_i\).
- Device \(v_{i,4}\) is at position \(\langle a_i, \sqrt{a_i/2 - 1} + 1 \rangle\) and has a maximal transmission range of \(\sqrt{2}\).
- Device \(v_{i,5}\) is at position \(\langle a_i + 1, \sqrt{a_i/2 - 1} \rangle\) and has a maximal transmission range of \(\sqrt{a_i/2 - 1}\).
- Device \(v_{i,6}\) is at position \(\langle a_i + 1, 0 \rangle\) and has a maximal transmission range of \(a^*/2\).
Figure 4.4.2: Complete instance from the reduction.

By construction, the only path from device $v_{i,1}$ to device $v_{i,6}$ within the gadget $i$ is via the devices $v_{i,2}$, $v_{i,3}$, $v_{i,4}$ and $v_{i,5}$, with costs $a_i^2 + a_i + 1$. Further, an increase of device $v_{i,0}$'s maximal transmission range from zero to $a_i$ makes it possible to go from device $v_{i,1}$ over device $v_{i,0}$ directly to $v_{i,6}$. The resulting costs of the latter path are equal to $a_i^2 + 1$.

These gadgets are sequentially put together along the $x$-coordinate (see Figure 4.4.2). The $x$-coordinate of each device in gadget $i$ is translated by $\sum_{j=1}^{i-1} a_j + (i - 1)(a^* + 1)$. Between each pair of gadgets $i, i + 1$ with $1 \leq i < |A|$, a further device is placed in the middle of the line segment between device $v_{i,6}$ and $v_{i+1,1}$ with a maximal transmission range of $a^*/2$. The devices $v_{1,1}$ and $v_{|A|,6}$ constitute the source-destination pair of the single commodity. To complete the construction, the parameter $R$ is set to $B$, and $Z$ to $|A|(a_i^2 + a_i + 1) + \frac{|A|-1}{2}(a^*)^2 - B$. This instance of RANGE CHANGE FOR SINGLE COMMODITY WITH BUDGET is computable in polynomial time.

It remains to prove that the subset sum instance is a YES-instance if and only if the corresponding transmission range problem is a YES-instance. A solution of the subset sum problem can be straightforwardly transformed into a solution of the transmission range problem. To do so, we increase the maximal transmission range of the devices $v_{i,0}$ with indices $i \in A'$ by $a_i$. The constraint on $R$ holds, and the cost of $SP(T')$ becomes exactly equal to $Z$. 
For the reverse direction, we first show that a transmission range change of any device \(v_{i,j}\) inducing a new path within gadget \(i\) is of no use, for \(i \in \{1, \ldots, |A|\}\), \(j = 1, \ldots, 6\). We start with device \(v_{i,1}\). If device \(v_{i,1}\) increases its maximal transmission range to directly reach \(v_{i,6}\), then the costs are equal to \((a_i + 1)^2\), and no shorter path is induced. A transmission range change to directly reach \(v_{i,3}, v_{i,4},\) or \(v_{i,5}\) does not help either, as the path over \(v_{i,2}\) is always shorter. Next, we investigate device \(v_{i,2}\). Sending directly to \(v_{i,4}\) has equal costs as sending via \(v_{i,3}\) to device \(v_{i,4}\). If \(v_{i,2}\) increases its maximal transmission range to reach \(v_{i,5}\), then its costs are

\[
\left(\sqrt{a_i/2 - 1} - 1\right)^2 + (a_i + 1)^2 = a_i^2 + \frac{5}{2} a_i - 2 \sqrt{\frac{a_i}{2} - 1} + 1.
\]

In contrast, the costs of the path along \(v_{i,3}, v_{i,4}\) are \(a_i^2 + a_i/2 + 1\). Since \(2\sqrt{a_i/2 - 1} \leq 2a_i\), no shorter path is induced. As a last option for device \(v_{i,2}\), it can increase its maximal transmission range to reach \(v_{i,6}\). Similar to the same case for device \(v_{i,1}\), the costs do not decrease with respect to the existing path. We continue with the device \(v_{i,3}\). For this device, it is always better to go via \(v_{i,4}\) to the device \(v_{i,5}\) respectively \(v_{i,6}\). The same holds for the device \(v_{i,4}\). Next, we show that inducing a new path between two consecutive gadgets does not reduce the shortest path cost either. Note that we only have to show that a transmission range change of device \(v_{i,5}\) or \(v_{i,6}\) is of no use, as a path from any other device to the next gadget is always more expensive than the path from the device over \(v_{i,5}\) or \(v_{i,6}\) to the next gadget. A direct path from device \(v_{i,6}\) to any device in the next gadget does not lead to a shorter path than via the device in between the two gadgets. If device \(v_{i,5}\) increases its maximal transmission range to directly reach \(v_{i+1,3}\), then the costs of this newly induced path are \((a^*)^2 + 1\). In contrast, the costs along the existing path via \(v_{i,6}\), the device between the gadgets \(i\) and \(i + 1\), \(v_{i+1,1}\), and \(v_{i+1,2}\) add up to

\[
\frac{a_i}{2} + \frac{a_{i+1}}{2} + \frac{(a^*)^2}{2} - 1.
\]

Since \(a^* = \max_i a_i\), the existing path is shorter. As the difference between a newly induced path from device \(v_{i,5}\) to \(v_{i+1,1}\) respectively \(v_{i+1,2}\)
and the existing path between the devices is only greater, a transmission range change does not induce a shorter path between consecutive gadgets. Lastly, an increase of the device in the middle of the gadgets does not induce a shorter path, because the path via $v_{i+1,1}$ is at least as short. We conclude that only transmission range changes of devices $v_{i,0}$ for any $i \in \{1, \ldots, |A|\}$ can yield a shorter path. To complete the proof, we assume that there is no solution for the subset sum problem instance. Then, consider any transmission range change in the constructed instance. Since the subset sum instance has no solution, any change at a subset of the devices $v_{i,0}$ either violates the budget constraint $R$ or the shortest path constraint $Z$. So, the transmission range change problem instance has no solution either.

\[\square\]

4.4.6 Transmission Range Change in a Subset

The variant where we control a subset $\tilde{V}$ of all devices is also of interest in the range change problem. The iteration algorithms naturally extend to this variant, and yield a running time $O(|\tilde{V}|(\log n + m)$ for the single device and single commodity case and a running time $O(|\tilde{V}|(\log n + m + k))$ for the single device and multiple commodities case. The expanded layer graph algorithms can be adapted to work in this setting as well. The set of directed edges between layers is reduced, as only vertices in the controlled vertex set $\tilde{V}$ have outgoing edges. More formally, a directed edge between vertex $(u, h)$ and vertex $(v, h+1)$ exists in the layer graph if $r(u) < |uv| \leq r(u) + r^+$, for $u \in \tilde{V}$, $v \in V$, and $0 \leq h < \Delta n + 1$. The $\mathcal{NP}$-hardness for the case with multiple devices and multiple commodities, as well as for the case with multiple devices, a single commodity plus a budget still holds by setting $\tilde{V}$ to the whole device set $V$.

4.5 Bundling Several Communication Requests

Until now, the cost of an edge is charged for every source-destination pair using it. In this section, we consider the problem where a device can bun-
dle the communication of several source-destination pairs, meaning that
the corresponding edge cost is charged only once. Formally, the decision
version of the problem is stated as follows.

**PROBLEM: BUNDLING TRANSMISSION COST**

**INSTANCE:** An instance $I = (T, Z)$ of BUNDLING TRANSMISSION COST
consists of a transmission graph $T = (V, E, K, p, r, c)$, and a posi-
tive number $Z$.

**QUESTION:** Is there a subgraph $T' = (V', E', K, p, r, c)$ with $V' \subseteq V$
and $E'$ based on $r$ such that each source-destination pair in $K$ is
connected and $\sum_{e \in E'} c(e)$ is at most $Z$?

Note that we concentrate on the version of the bundling transmission
cost problem without any modification operations. We are only interested
in finding cheap routes between source-destination pairs within the existing
setting.

The bundling transmission cost problem is a geometric version of the
so-called point-to-point connection problem with fixed destinations [53].
In the point-to-point connection problem with fixed destinations, we are
given a graph $G = (V, E)$ with nonnegative costs $c(u, v)$ for each edge
$(u, v) \in E$, and source-destination pairs $\{(s_1, t_1), \ldots, (s_k, t_k)\}$. The goal
is to choose a subset of edges such that all source-destination pairs are
connected and the total cost of the chosen edges is minimized. If we choose
$s_l = s$ for all $l \in \{1, \ldots, k\}$, then the point-to-point connection problem
is equivalent to the Steiner tree problem, and thus $\mathcal{NP}$-hard.

**Theorem 4.16.** BUNDLING TRANSMISSION COST is $\mathcal{NP}$-hard.

**Proof.** The problem is the special case of the point-to-point connection
problem with fixed destinations where the underlying graph is a Euclidean
graph and the edge costs are equal to the squared distances between the
adjacent vertices. Since the Steiner tree problem is also $\mathcal{NP}$-hard for the
setting where the edge costs equal the distance between the adjacent vertices raised to a power greater than one [14], the bundling transmission cost problem is \( NP \)-hard as well.

\[ \square \]

### 4.6 Conclusions and Open Problems

We analyzed the complexity of many strategic device operation problems in this chapter. The problem variants with multiple devices and multiple commodities are proven to be hard for all operations, and the problem variants with either a single device or a single commodity, for which the complexity is known, are solvable in polynomial time. It would be interesting to answer the complexity status of the problem variants which are still open.

Another open issue is the combination of different operations. A problem of this kind would for instance be the following: given a transmission graph \( T \), a positive integer \( \Delta n \), and two positive numbers \( r \) and \( r^+ \), change the maximal transmission range of \( \Delta n' \) devices by at most \( r^+ \) and place \((\Delta n - \Delta n')\) additional devices with maximal transmission range \( r \) such that the profit is maximum. Similar to Chapter 3, it is also interesting to study the game arising when several players simultaneously or sequentially delete their devices, move them, or change their maximal transmission range.

A further interesting research direction is to derive faster algorithms by exploiting the fact that the points lie in the plane, and that the costs are related to the Euclidean distance.
Bibliography


Bibliography


Curriculum Vitae

Luzius U. Anderegg

date of birth: August 25, 1974
place of birth: Zurich, Switzerland
citizenship: Switzerland

Education:
June 2001 – June 2006: ETH Zurich, Zurich, Switzerland
PhD student at Institute of Theoretical Computer Science with Prof. Dr. Peter Widmayer
academic title: Dr. sc.

October 1993 – September 1999: ETH Zurich, Zurich, Switzerland
student at Department of Computer Science with minor Operations Research
academic title: M. Sc. (Computer Science), ETH

April 1987 – October 1993: Realgymnasium Raemibuehl, Zurich, Switzerland
high school
degree: Matura (Swiss high school diploma)

Work experience:
February 2000 – May 2001: IBM Switzerland, Zurich, Switzerland
Member of the Business Intelligence and CRM group of IBM Switzerland

October 1995 – September 1999: UBS, Zurich, Switzerland and EDIFACT, Duebendorf, Switzerland
part-time work at the IT department
Publications:

Refereed Proceedings


Technical Reports


Book chapter