Using Mobile Relays for Ensuring Connectivity in Sparse Networks

Dissertation

by

Jarosław Kutyłowski

Fakultät für Elektrotechnik, Informatik und Mathematik
Institut für Informatik und Heinz Nixdorf Institut
Universität Paderborn

Paderborn, September 2007

Reviewers:

- Prof. Dr. Friedhelm Meyer auf der Heide, Universität Paderborn, Germany
- Prof. Dr. Christian Schindelhauer, Universität Freiburg, Germany
- Prof. Dr. Sándor Fekete, Technische Universität Braunschweig, Germany

With great thanks to my wife Iwonka
for her support during the tougher moments

Acknowledgements

I am much obliged to my supervisor, Friedhelm Meyer auf der Heide, for providing excellent support throughout the three years working at this thesis. Many of our long discussions led to great ideas, or at least to the exclusion of bad ones.

Further thanks goes to Christian Schindelhauer for his engagement in the work on Smart Teams in the early stage of the project's emergence. It's been also this early time of my university work when Mirek Korzeniowski and Marcin Bieńkowski welcomed me in Paderborn and showed that finishing a thesis is actually possible. They have also contributed to the creation of this thesis, by proofreading it thoroughly.

I don't want to forget the whole "Algorithm and Complexity" working group and especially Mirek Dynia for sharing the lows and highs of the daily work, fruitful discussions and entertaining moments.

Last but not least I owe thanks to my parents, who have provided me with an inquiring view of the world and encouraged to taste the challenges of academic life.

Contents

CHAPTER 1

Introduction

Disaster-struck areas on earth or the surface of other planets seem to be a natural deployment environment for robots, as humans can hardly survive in the conditions encountered there. Many examples of such applications can be shown in today's technology: exploration of planets relies heavily on robots [Mat96], emergency crews are using them when searching for survivors [Mur04], etc. Current state-of-the-art focuses on employing one or few robots to perform these tasks. On the other hand, a widely spread belief is that this tasks can be better performed by large groups of self-organizing simple robots used in replacement for few complicated ones. A common term used in literature to describe such groups is robot swarms; we will stick to using the term *robot team*. Using robot teams has certain advantages: they are cheap, one expects the failure of a fraction of them to be perfectly tollerable and one can expand the possibilities and the performance of a team by scaling up the number of robots used.

The employment of massive scale robot teams also implies several problems: as in the case of traditional parallel computational systems, coordination and communication become extremely important for the performance of the parallel system as a whole. This is even more important in the case of robot swarms than in parallel computational models, both due to the large number of participants in the network and due to the slow wireless connections expected to be used in robot swarms. Therefore, it is necessary to employ local strategies, in which robots have only a limited knowledge of their environment and do not necessarily know the state of the system as a whole.

This thesis works out one of the most important issues needed for self-organization of large robotic teams. In more detail, we are investigating means for providing stable communication for wireless networks composed of robots, dispersed scarcely in vast terrain. Our main concern is ensuring connectivity between network parts which are far apart. In particular, we concentrate on mechanism which actively counteract the disconnection of the network in presence of dynamics. For that purpose we use relays,

i.e. mobile robotic devices, which can forward communication messages along certain routes. We are aiming at providing a background communication service and therefore do not pose any limiting assumptions on the movement of the network as such – we aim at providing a service which keeps the network connected under any possible dynamics of the network.

Solving the described problem is a challenging task, which we have separated in this thesis into three separate problems. Before we go more into detail on these subproblems, let us define the general assumptions we are taking into account.

Basic Model for Wireless Communication. In a wireless communication network we are usually given a set of nodes equipped with wireless transceivers, which may use one of the available transmission mediums, e.g. radio waves, infrared, to communicate with each other. For the purpose of the algorithm design and analysis one has to abstract from the actual physical network and develop a (mathematical) model of the wireless network.

Usually strong simplifying assumptions are made in order to create a model which has a complexity one can handle. There are two main types of restrictions: such that determine the environment and thereby affect the possible positions of the nodes and such that define the communication abilities of the nodes. For the first type, the great majority of research models the environment of the network as a plane. If it is required by the application in mind, one may restrict the positions the nodes can attain on this plane by introducing obstacles. These obstacles correspond to areas on the plane which nodes cannot enter. For the second type of restrictions, it is very often assumed that two nodes may exchange information in a wireless network if and only if they are in some constant distance. This is because the wireless transmission media used have a radial propagation (ignoring the existence of directed antenna). Therefore wireless communication can be thought of as a local broadcast in a bounded area. The communication graph of the network is then defined to contain one node for each node of the wireless network and an edge between two nodes if and only if the corresponding nodes in the wireless network are within communication distance. This kind of graph is called a disc graph [CCJ90], as nodes are able to communicate only if inside of discs around their neighbors' positions. The disc graph model is often argued, not without reason, to be very imprecise. Though, as we are more concerned about high-level issues of organizing the communication and do not want to go into technical details of communication between neighbored nodes, the disc graph model is despite all of its deficiencies a reasonable abstraction in our situation.

In this thesis we will think of a wireless network as of a graph embedded on a plane either containing obstacles or not. As we will be primarily concerned with mobility of

nodes in the wireless network, the graph will change in time. Therefore, we will say that the graph is dynamic.

General Idea. Recall that we want to deal with sparse networks dispersed on a large area. In such networks distances between network parts may increase above the communication distance of the wireless communication mechanisms used. Then, there is no possibility for direct communication between nodes in different parts of the network. We propose a solution which ensures that the network remains connected even if the described situation occurs. This solution relies on the application of mobile relays – these relays are actively directed in the terrain so that they form communication paths between network parts. The relays forming such a communication chain are expected to be in a distance allowing wireless communication with their neighbors. Therefore, they are able to route messages between network parts which would have been disconnected otherwise. Such a chain of relays will be called *communication chain* throughout the rest of this thesis.

This idea of forwarding messages along a communication chain between distant places is not necessarily new. The humankind has already used camps on mountains to forward messages with light or smoke signals on long distances. In this thesis we investigate the application of this idea in a robotic setting with mobile relays.

Challenges. Consider a graph representing the network established by the robot teams in the terrain, such that each node corresponds to a robot and an edge connects two nodes if and only if there is a direct wireless connection between the two corresponding robots. This *network graph* may have several distinct connected components and, as already explained, our goal is to have these connected components connected.

In our scenario the mobile relays are expected to be robotic devices. The number of such relays available at our disposal in a network will be limited. Therefore, we should be able to use as few of the relays as possible for reaching our goal of connecting the network graph.

We can reformulate our goal saying that we are looking for appropriate positions for relays, such that using the formed communication chains the network graph becomes connected, while minimizing the number of used relays. Since we do not know anything about the future movement behavior of the robots which compose the network (or at least have only very limited knowledge about it), we cannot predict the future layout of the network graph. Therefore, we have to develop strategies which allow the relays to adapt to changes in the graph on-the-fly.

While coping with the mobility of the network, we want not only to use the least possible number of relay nodes for maintaining connectivity but also save their energy

used for movement as far as possible. This allows a system composed of relays to operate resource-efficient.

As we are interested in a large, wide-spread networks we aim at developing local and distributed strategies. Thus, ideally, the relays should base their decisions on observations of their direct environment or information communicated by relays in their neighborhood.

Compacted robot graph. In a great part of this thesis we will not be interested in separate robots which constitute the connected components of the network graph; rather we want to consider only the connected components. We thereby construct the *compacted robot graph* by contracting each connected component into one node.

The edges of the compacted robot graph are intended to represent possibilities for communication chains to be set up. Therefore, an edge connects two nodes of the compacted robot graph if there is a path in the terrain, such that we can set up a chain of relays on this path. The weight of the edge corresponds to the length of the path and thereby expresses how many relays are needed to construct a communication chain on this path. We will say that a communication chain *supports* a path, when the relays have been set up appropriately to form a communication chain along this path, connecting the two network parts on the endpoints of this path.

We define the *support graph* to be a subgraph of the compacted robot graph, contain all of its nodes and only those edges, which are supported by a communication chain. We will say that the compacted robot graph is *fully supported* if the support graph is connected.

The compacted robot graph is dynamic in that the edge weights may change with time. This corresponds to robot teams in the original network graph having moved. We assume no nodes/edges are added or removed to the compacted robot graph, which would reflect connected components of the network graph splitting.

1.1. Scope

The definition of the compacted robot graph results in a natural separation of our general problem into two levels. On the lower level, we have to take care that relays can organize to a single communication chain on a path between two nodes of the compacted robot graph. On the higher level, we have to decide which paths have to be supported by the communication chains at all, so that the robot graph is fully supported. This separation will guide the division of this thesis in two parts.

In the first part we will deal with the question on how relays can organize on the plane to form a communication chain with a length as short as possible. In the second part we will concentrate on two questions. First, we want to know which paths to use in a dynamically changing compacted robot graph, so that the energy used by the relays for moving between different paths is minimized. The second question deals with the determining of an assignment of free relays to communication chains.

In the next three paragraphs we go more deeply into describing the basic challenges in answering each of those issues.

Organizing relays to a communication chain. The first question relates to a basic problem, in which we want to connect two points on a plane with a communication chain. We consider a situation with one stationary point, called the *base camp* and a mobile one, the *explorer*. The explorer and base camp are obviously abstractions for whole groups of robots, i.e. nodes of the compacted robot graph.

We pose no restrictions on the mobility of the explorer other than restricting its maximum speed. Therefore the relays have to maintain a communication chain without any information about the future movement of the explorer. This can be a challenging task, if we pose some restrictions on the abilities of the relays. In the simplest case we model the robots functioning as relays as oblivious, possessing no memory and basing their current action only on the positions of their neighbors. We investigate the same problem also for extended types of robots, which have some memory and access to a GPS-like system. Furthermore we investigate the influence of different positioning methods and imprecise measurements on the performance of relays.

In our solutions we want to minimize the number of relays used for the chain while we assure that the chain is always connected. Furthermore, we investigate whether and how the behavior of the communication chain obstructs the movement freedom of the explorer.

Minimizing number of changes in support graph. We can place communication chains in different ways in the compacted robot graph to make it fully supported. Since we are assuming that the number of relays available is limited, we are looking for locations for communication chains such that the total length of the chains is minimized, while the robot graph is still fully supported.

In a static setting, with the robot graph not changing this is quite simple – taking any minimum spanning tree as the support graph for the compacted robot graph we have a solution with the minimum number of relays used.

In the dynamic setting with a changing robot graph the choice of a proper minimum spanning tree is not that easy any more. As the minimum spanning tree constituting the

support graph is changed by our strategy, relay stations have to travel between different paths in the terrain. This implies energy cost, which we want to minimize. It is therefore not sufficient to recompute the minimum spanning tree every time a substantial change in the robot graph occurs. We want our strategies to also maintain similar spanning trees for a long period of time, decreasing the amount of energy consumed by relays for traveling.

Managing spare relays. The third problem investigated concerns the management of spare relays. We assume that the robot graph is held connected by relays, which support distinguished communication paths. The number of relays necessary to support a communication chain is proportional to the distance between the connected components it connects, as the relays have to stand in some fixed distance to each other.

Since the compacted robot graph changes in time, communication chains change their length. Therefore the required number of relays to support these paths changes in time. When the length becomes smaller, some of the relays used up to now are unnecessary. This generally yields no problems, a more problematic situation occurs when a communication chain becomes longer and no spare relays are available next to this communication path – then it is necessary to fetch a relay from a more distant position to the communication chain. This introduces energy cost for the relay which has to travel to its new destination. If caring about energy usage, one would try to use the relay which is available in a local surrounding. On the other hand, such a greedy strategy may have a very bad worst-case performance.

1.2. Outline of the Thesis

Part I of this thesis deals with the organization of communication chains by relays. We investigate several strategies, described in Chapters 3, 4, and 5. In Chapter 2 we give a general introduction to the topic handled in Part I.

In Chapter 6 of Part II we continue with the discussion on saving energy usage of relays by minimizing the number of changes between supporting different paths in the compacted robot graph. In Chapter 7 we discuss how to manage spare relays.

As we are dealing with essentially three different topics in this thesis, the notation is separate for each of them. There are some similarities in notation between the chapters of Part I, which we introduce in Chapter 2. For convenience, we will also delay introducing formal models and state related work closely connected to the subproblems to each of the chapters.

1.3. General Related Work

As we have already described, this thesis basically consists of three problems. Although the three problems integrate into one architecture, they are intrinsically different. Therefore they are handled separately in the respective chapters together with the relevant related work. In this general overview of related work we only refer to literature which is relevant to this thesis as a whole.

The connectivity of wireless networks has been already considered from various points of view. A wide range of works is concerned with the connectivity of randomly generated networks. So, in [SB03, GK98] the authors investigate the necessary transmission ranges to obtain connectivity in such a graph. Similarly, one can investigate the number of neighbors to which a node should be connected in order for the whole network to be connected [XK04].

The idea of using mobility to improve the capacity and connectivity of networks has been already investigated in [GT01, Hor07]. In this case the movement of nodes is not influenced directly, rather the mobility typical for the behavior of the wireless network is used in order to improve the parameters of the network. In [SB03] the influence of mobility on the required transmission range for random networks is also investigated.

The novelty of this thesis lies in the active usage of mobility in order to guarantee connectivity in a scenario with adversarial and not stochastic dynamics. A similar idea of using relays for forwarding communication in a network has been only recently presented in [NFP+03, NPGF04] where the authors are using relays to connect a robot to a base camp. In contrast to this thesis, their work concentrates on engineering aspects of this problem.

1.4. Bibliography Note

Many of the results presented in this thesis have been already presented and published in a preliminary version in conference proceedings.

The general model used in this thesis and the problem of organizing relays to chains has been first presented in [DKLM06]. The work contains first theoretic results on the *Go-To-The-Middle* strategy in the static scenario. The investigation of this strategy has been extended in [DKMS07], which also contains the results from Chapter 5. The remaining results presented in Chapter 3 and 4 have not been published yet.

A large part of the work on minimum spanning trees to be found in Chapter 6 has been published in [DKK07]. The problem described in Chapter 7 has been introduced and investigated in [BK07].

Part I.

Organizing a Communication Chain

CHAPTER 2

Communication Chain Problem

In the first part of the thesis we will deal with the question on how to organize a single chain of relays connecting two points on the plane. In this chapter we will introduce the models used afterward in the remaining chapters of this part. We furthermore give a rough overview of the results and compare them.

In general we assume that the chain of relays has to connect two points on the plane, one of which is stationary and the second one is mobile. For the sake of clarity we name the stationary point the *base camp* while the mobile point is named *explorer*. The relays are mobile and their movement can be directed freely by strategies executed by those relays.

Up to now we have informally spoken of a *chain* of relays. Hereby we mean an ordered sequence of relays, denoted by $v_1, \ldots, v_n$. For technical reasons we include the explorer and the base station into the chain, as respectively its first and last element. Thus, v_0 describes the explorer and v_{n+1} the base station. Elements of the chain will be referred to as *stations*, meaning either a relay, the explorer or the base camp. The main restriction on the chain is that we require the distance between stations neighbored in the chain to be limited to the transmission distance denoted by $\mathcal{R}$. Thanks to this, the chain can forward communication messages between the base camp and the explorer.

We want to design strategies for the relays, which guide their movement on the plane. Each of the relays is assumed to execute its own copy of this strategy and is able to sense the environment. We will go deeper into detail on the computational and sensing abilities of the relays when describing the relay model later.

Requirements on the Chain

The movement pattern of the explorer is assumed to be unpredictable except that we assume it has a maximum movement speed. Therefore the primary goal is to organize the chain in such a way, that the distance between any two neighbored stations in the

chain does not exceed the distance $\mathcal{R}$. Therefore, the strategies employed by the relays have to ensure that their positions follow the movement of the explorer.

A secondary goal for the strategies is to minimize the number of relays used in the chain, i.e. arrange the relays near to the direct line connecting the explorer and the base camp.

The greatest challenge for this problem lies in the fact that we are looking for strategies which operate locally and with very simple logic. Therefore, we have to design a solution in which each relay computes its position on its own, where communication is not used at all and relays even posses no memory. We are interested to see whether one can design efficient strategies solving this problem with this simple model and what are the parameters which hinder an efficient solution.

Terrain Model

The relays are moving on a terrain represented by a plane. Both the explorer and the relays may move freely. In this setting, the relays should (in an ideal case) arrange on a line segment connecting the base camp and the explorer. With relays placed on this line segment in distance $\mathcal{R}$, the chain uses the least possible number of relays to connect the base camp and the explorer.

For technical reasons, we assume that there may be more than one relay present at one point of the terrain.

Time Model

We assume that the execution of the strategies of all relays is synchronized. Therefore, we can divide the time in discrete time steps. We require the execution of a strategy in a time step to fit into the *LCM*-model (as in [CP04b]). In this model, the execution of each step is divided into three operations. These operation are: *Look*, *Compute*, *Move*. In the first operation of the step the relay gathers new information about its environment by scanning it with its sensors. In the second operation, the sensoric input obtained in the previous time step is analyzed and a decision is made about the behavior of the robot within the current step. During the *Move* operation, the robot moves to a position precomputed in the *Compute* operation.

As we divide the time in discrete time steps, we are assuming that the relays involved in the chain are synchronized. That means, that a robot finishing moving earlier then others waits until the rest is finished (this may be made explicit by inserting a *Wait* operation into the definition of a step).

Examining strategies in the *LCM*-model implicitly assumes that one regards a *LCM*-step as an atomic time unit and determines the performance of strategies basing on the number of such steps necessary to reach a certain condition. This implies that a *Look*-operation is accounted a comparable amount of time as a *Move*-operation. In order not to keep this assumption realistic, we will be restricting the maximum distance a relay can move during one time step. On the other hand, in contrast to our intuition, the *Move* operation is not necessarily that one which takes the longest time. Very often, gathering the sensoric input is even more time-consuming, if for example laser scans of the surrounding must be taken. An example of such a scenario can be found in [FKN06].

Relay Model

Throughout the next chapters we will investigate the problem of maintaining a chain of relays with respect to different models of relays.

Memory. Relays possessing memory will be called stateful, while those without any memory will be described as stateless. Stateless relays have to base in the *Compute*-operation only on readings from the current *Look*-operation, without any knowledge on their own historical behavior.

Local coordinate systems. We assume that each of the relays is able to construct a view on its environment by using its sensing abilities. For all strategies presented in the following chapters, the position of a relays' neighbors in the chain play the most important role. Therefore, each of the relays can be thought of as constructing its local coordinate system with its own position and the positions of its neighbors marked. This local coordinate system is constructed basing on sensor input which can be of different quality. We can assume the distances and relative bearings of a relays' neighbors be measured with perfect accuracy, providing it with a precise local coordinate system. If these local measurements are imprecise, we have to deal with an imprecise local coordinate system.

Relations between local coordinate systems. Another important issue is the relation of the local coordinate systems of different relays to each other. On one hand, we can assume that local coordinate systems have nothing in common, i.e. two local coordinate systems may be rotated and translated arbitrarily with respect to each other. This corresponds to a setting where a relay can only measure the distances and relative bearings between itself and its neighbors.

Then, the local coordinate systems may be rotationally aligned, all sharing a common notion of X and Y axis. Once again, the rotational alignment may be precise or imprecise, depending on whether we allow a bounded error in the alignment. This kind

of rotational alignment of local coordinate systems may be accomplished by providing relays with a compass.

Furthermore, the local coordinate systems may be also aligned with respect to their origins, which can be accomplished by using a positioning scheme similar to GPS, which provides each relay with an absolute position. This is obviously the strongest setting.

Performance Measures

The analysis of our strategies will be performed in two settings: either static or dynamic. In the static case we will be given a fairly arbitrary chain configuration on the plane at the beginning. While the explorer won't be moving, we will investigate the time needed for the relays to become arranged approximately on the line between the base camp and explorer.

In the dynamic case, we will start with some well-defined chain configuration and let the explorer move. We will then examine the performance of the strategy in dealing with the explorer's movements. In this scenario we measure the required speed for relays in comparison to the speed of the explorer and the length of the chain maintained by the strategy.

Although the dynamic scenario is of more importance for practical reasons, the results from the static scenario can be used for comparing the performance between different strategies and in order to understand the weaknesses of strategies.

Lower Bound

Any strategy which solves the communication chain problem in the static scenario requires at least $\Omega(n)$ time steps in order to organize the chain approximately on the line segment between the explorer and the base camp. This can be intuitively seen, as the relays with large distance to the line segment between the base camp and explorer either have to decrease this distance by $\Omega(n)$ or the decision to remove them must be propagated through the chain. Both takes at least $\Omega(n)$ time steps.

In the dynamic scenario it is necessary for the relays to move with a speed at least as large as the explorer's maximum speed. This poses a lower bound of 1 on the ratio between relay and explorer maximum speed.

Outline

The contents of this part are divided into three chapters. In Chapter 3 we introduce the *Go-To-The-Middle* strategy. It is a very simple and natural strategy, which works with stateless relays with precise local coordinate systems, which do not have to be aligned in any way. As we will show that simplicity sacrifices performance. Furthermore, we

will be able to show that the strategy does not work correctly when the local coordinate systems are imprecise.

In Chapter 4 we present the *Hopper* strategy, which is optimal up to constant factors in both the static and dynamic scenarios. The presentation of the *Hopper* strategy will be preceded by the introduction of a simpler *Manhattan-Hopper* strategy which makes extensive use of an alignment of stations on the plane in a grid structure to improve performance. *Manhattan-Hopper* and *Hopper* rely on sequential execution of the strategy in runs. For this purpose, relays have to be able to signal their neighbors that they are ready with executing their action. This is different from the other strategies presented in this thesis, which are executed by all relays in parallel.

Both the *Go-To-The-Middle* and *Hopper* strategies can work with local coordinate systems of relays independent of any global coordinate system. Therefore, the local view of the relays on their environment is completely sufficient in order to derive their next actions. In Chapter 5 we introduce the *Chase-Explorer* strategy, which achieves a performance even better than that by *Hopper*. Nevertheless, it requires relays to memorize the position of the base camp. In its original form, it requires local coordinate systems to be aligned both rotationally and with respect to their origins.

As we will explain in more detail in Chapter 3, the *Go-To-The-Middle* strategy, in contrast to *Manhattan-Hopper*, *Hopper* and *Chase-Explorer*, requires additional mechanisms to insert and remove relays from the chain, which is done in an easy and intuitive way in the remaining strategies.

The following table gives a compact overview of the performance of each of the strategies. The *static* column gives the time necessary for the relays to be organized to a nearly-optimal chain in the static scenario in the worst-case. The *dynamic* column describes the ratio of the required relay speed to the allowed speed for the explorer.

strategy	**relay model**	**GPS**	**static**	**dynamic**
Go-To-The-Middle	stateless	no	$O(n^2 \log n)$	$O(1/n)$
Manhattan-Hopper	stateless	no	$4n$	$(1 + \sqrt{2})$
Hopper	stateless	no	$25n + 1$	$O(1)$
Chase-Explorer	stateful	yes	$\mathcal{R}n$	1

Related Work

A variety of problems similar to the maintenance of a communication chain have been investigated in the area of swarm robotics. Maintaining a communication chain means that relays are self-organizing to form a geometric figure, namely a a line, on the plane.

Therefore, we will shortly survey previous research which deal with ordering robots to form some geometric figures on the plane.

One of the basic problems investigated in this setting is the gathering problem [Mat92]. A dispersed group of robots, represented by points on a plane, is required to gather at a point in the plane. This point can be chosen freely by the robots, it is only important that all robots finally gather at one point. It turns out that in an asynchronous model, in which the duration and the starting/ending of time steps may vary between robots, it is impossible for deterministic robots to gather at one point in finite time [CP02]. Allowing more than 5 robots to detect whether multiple robots occupy one point allows to gather at one point in finite time [CFPS03]. The problem may be also considered in a setting with robot failures [AP04]. The problem turns out to be slightly easier in the semi-synchronous model, in which the time steps are synchronized among robots but robots may be put to sleep during any time step. Gathering is then possible with already 3 robots in finite time [SY99]. Recently the problem has been evaluated in the asynchronous setting with robots equipped with imprecise compasses [KTI+07].

The convergence problem is the easier variant of the gathering problem, in that we only require all robots to converge in infinite time to one point. The gravitational algorithm which lets are robots move toward their center of gravity converges in the synchronous and semi-synchronous model [CP04b] and also in the asynchronous model [CP05]. The problem can also be solved in the semi-synchronous model when the sensoric input and the odometry of robots is imprecise [CP04a]. A different model with robots having only limited visibility of the plane has been shown to be solvable in [AOSY99].

The problem of pattern formation is slightly more involved, as robots are required to form a pattern on plane, rather than only moving to one point. Asynchronous robots can form any in advance known pattern if they know the orientation of two axes, while it can be only solved for an odd number of robots when only the orientation of one axis is known [FPSW99]. Certain patterns can be formed even without knowing a common orientation, depending on the starting positions of the robots [SY96]. In [DK02] and [CMN04] algorithms for forming a circle by robots in the semi-synchronous model are evaluated.

Notation

Generally, we will be investigating our strategies with a time step being an atomic unit of time. Then the number of relays in the chain changes with time and is denoted by n_t. Recall that we denote stations in the chain by $v_0, \ldots, v_{n_t+1}$, where v_0 denotes the explorer and v_{n_t+1} the base camp. By $p_t(i)$ we define the position of station v_i at the beginning of time step t. We set $b := p_t(n_t + 1)$ to be the position of the base camp.

The coordinates of a point p in a cartesian coordinate system will be denoted by $(x; y)$. A line segment or a line defined by two points p, q will be denoted by $\langle p, q \rangle$. The distance of the explorer to the base station in time step t will be often described shortly as $d_t = |p_t(0) - b|_2$. The notation $|x|$ denotes the absolute value of x, whereas $p - q|_2$ is the Euclidean distance between the points p and q.

CHAPTER 3

Go-To-The-Middle Strategy

In this chapter we will introduce and analyze the *Go-To-The-Middle* strategy. The *Go-To-The-Middle* strategy assumes a very limited model of stateless relays, each having a separate local coordinate system not aligned with others. Throughout the whole chapter we will assume that the local coordinate systems are precise, i.e. the relay is able to measure the position of its neighbors precisely. Only in Section 3.5 we will show that with imprecise local coordinate systems the *Go-To-The-Middle* strategy is not guaranteed to work properly.

The *Go-To-The-Middle* strategy works as follows. In the *Look*-operation of a time step t the relay v_i executing the *Go-To-The-Middle* strategy observes the positions of its two neighbors. Afterward it computes the position in the middle of the line segment $\langle p_t(i-1), p_t(i+1)\rangle$. During the *Move*-operation the strategy guides the relay to the computed middle of the line segment between the positions of its neighbors.

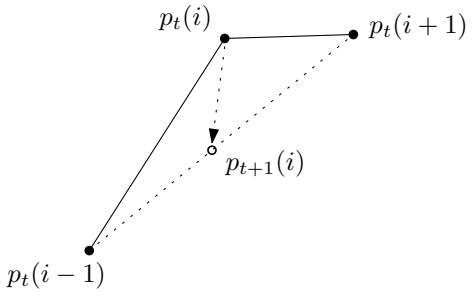


Figure 3.1.: Relay v_i executes the *Go-To-The-Middle* strategy

As the explorer moves, the distance d_t may change. Therefore the number of relays within the chain should be changed appropriately, so that the number of stations employed in the chain is neither too large nor to small. For the sake of clarity, we won't discuss that issue now and come back later to this topic in Section 3.4.

Explorer Behavior

We bound the maximum speed of the explorer to one distance unit per time step. Therefore we also require relays to be able to move for one distance unit per time step.

When the *Go-To-The-Middle* strategy is applied by relays, we require the explorer to carefully watch the distance between its position and the position of v_1. The explorer is responsible for not exceeding a distance of $\mathcal{R}$ to v_1 – even if that means it has to slow down or stop moving for some period of time.

We assume the explorer to move in parallel to the relays during the *Move*-operation.

Correctness/Performance Criteria

In order to organize the chain properly, the *Go-To-The-Middle* strategy must ensure that neighbored relays are always in distance at most $\mathcal{R}$ in the chain. As already noted, the explorer is responsible for ensuring that $\left|p_t(0) - p_t(1)\right|_2 \leq \mathcal{R}$. We will say that a communication chain is *communication-valid* in time step t if $\left|p_t(i) - p_t(i-1)\right|_2 \leq \mathcal{R}$ for all $i = 1, \ldots, n_t$.

Furthermore, in order not to exceed the movement capabilities of the relays, we want the *Go-To-The-Middle* strategy to dictate relays to move by at most unit distance per time step, formally $\left|p_t(i) - p_{t+1}(i)\right|_2 \leq 1$. If the strategy fulfills this requirement we call it *movement-valid*.

The performance of the *Go-To-The-Middle* strategy can be measured in two ways. In the static case we will be given a communication-valid chain at the beginning. While the explorer won't be moving, we will investigate the time needed by the chain to organize itself to a line connecting the explorer and the base camp. Here the convergence time will be given in the number of steps in the *LCM*-model.

In the dynamic case, we will assume a moving explorer. As already outlined, the explorer may be required to slow down if the station v_1 is not able to catch up with its position. Therefore, we will be investigating bounds on a guaranteed speed of the explorer.

Outline

We start our investigation of the *Go-To-The-Middle* strategy in Section 3.1 with showing that it organizes the chain of relays properly, i.e. that it is communication-valid and movement-valid.

The behavior of the strategy in the static scenario is analyzed in Section 3.3 and afterward in the dynamic scenario in Section 3.4. We quickly summarize the results obtained for the *Go-To-The-Middle* strategy in these two scenarios. In the static one, we are given any communication-valid chain with n relays and a stationary explorer.

We show that it takes $O(n^2 \log n)$ time steps to bring all relays in a constant distance to their optimal positions on the line between the explorer and base camp (Theorem 3.10). Complementary to the previous result, in Theorem 3.6 we can show that there exist communication-valid chains such that the relays need at least $\Omega(n^2)$ time steps to come near to their optimal positions.

In the dynamic scenario we are dealing with a moving explorer and are investigating how the *Go-To-The-Middle* strategy limits its speed. We start with a short discussion on ways to control the number of relays in the chain when the explorer changes its distance to the base camp. Unfortunately, the *Go-To-The-Middle* strategy needs certain extensions in order to be able to insert and remove relays from the chain.

We are able to show matching lower and upper bounds of $\Theta(1/d)$ on the average speed of an explorer in distance d to the base camp (Theorem 3.11 and 3.15). This result shows, that the speed with which the explorer can move is inversely proportional to its distance to the base camp – clearly a poor result for the *Go-To-The-Middle* strategy. Note that this is a worst-case guarantee – with an explorer which moves changing its direction very often the *Go-To-The-Middle* strategy is expected to perform much better.

Afterward, we show in Section 3.5 that the *Go-To-The-Middle* is not communication-valid if we are given an imprecise local coordinate system, where the sensor readings are erroneous.

Eventually we deal in Section 3.6 with strategies similar to *Go-To-The-Middle* and show that their performance cannot be essentially better than that achieved by *Go-To-The-Middle*.

3.1. Correctness

First, we want to show that the *Go-To-The-Middle* strategy is correct with respect to the criteria we have set up earlier. This requires that the chain we are given at the beginning is communication valid and that during the first time step no relay moves by more than 1. If these basic properties are fulfilled, the *Go-To-The-Middle* strategy is able to behave correctly, as shown in Theorem 3.1 and 3.2.

Theorem 3.1. *Assume that the explorer ensures that* $\left|p_t(0) - p_t(1)\right|_2 \leq \mathcal{R}$ *for every time step* t *and that the chain configuration is communication-valid at the beginning of the first time step. Then the chain configuration is communication-valid after every time step.*

Proof. We will show that the chain configuration is always communication-valid by induction on the time step. By assumption the chain is communication-valid at the

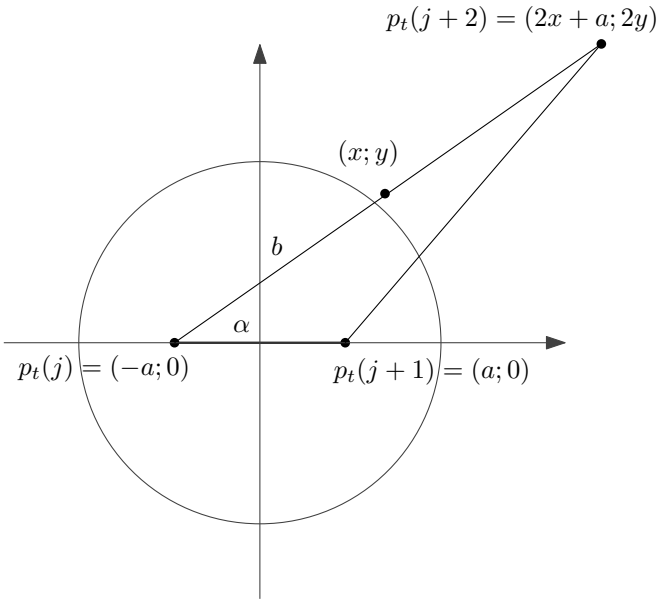


Figure 3.2.: $(x; y)$ is the new position of vertex $(a; 0)$ after applying *Go-To-The-Middle*.

beginning of the first time step. Now assume that at the beginning of time step t it holds $\left|p_t(i) - p_t(i+1)\right|_2 \leq \mathcal{R}$ for all $i = 0, \ldots, n_t$. We will show that $\left|p_{t+1}(j) - p_{t+1}(j+1)\right|_2 \leq \mathcal{R}$ after *Go-To-The-Middle* has been applied. Note that by assumption we have $\left|p_{t+1}(0) - p_{t+1}(1)\right|_2 \leq \mathcal{R}$ so that we do not have to care for the distance between the explorer and v_1.

Fix a line segment $e := \langle p_t(j), p_t(j+1)\rangle$, for any $j = 1, \ldots, n_t$. Let us lay out a coordinate system so that e lays on the X-axis and that the origin of the coordinate system is in the middle of edge e. Let $a := |e|_2 / 2$.

We draw a circle around the origin of the coordinate system with a radius of $\mathcal{R}/2$ as shown on Fig. 3.2. We will show that after applying *Go-To-The-Middle* both $p_{t+1}(j)$ and $p_{t+1}(j+1)$ lie within this circle. Then obviously the distance between them is at most $\mathcal{R}$.

Define $(x; y)$ to be the coordinates of $p_{t+1}(j+1)$ and $b := |p_t(j) - p_{t+1}(j+1)|_2$ to be the distance between the point $p_t(j)$ and $p_{t+1}(j+1)$. Let α describe the angle at point $p_t(j)$ between the lines $\langle p_t(j), p_{t+1}(j+1)\rangle$ and $\langle p_t(j), p_t(j+1)\rangle$. From an obvious geometric property we have $b = \sqrt{(x+a)^2 + y^2}$.

Since $p_t(j+1) = (a; 0)$ has been moved to $(x; y)$ by *Go-To-The-Middle*, we know that $p_t(j+2)$ lies on the line connecting $(-a; 0)$ and $(x; y)$ in distance b from $(x; y)$. From the triangle similarity property we have that $p_t(j+2) = (2x+a; 2y)$.

Now assume for the sake of showing a contradiction that $(x; y)$ lies outside the circle. Then $x^2 + y^2 > (\mathcal{R}/2)^2$. We now compute the distance $\left|p_t(j+1) - p_t(j+2)\right|_2$. We have

$$\left|p_t(j+1) - p_t(j+2)\right|_2 = \sqrt{(2x+a-a)^2 + (2y-0)^2} = \sqrt{4 \cdot (x^2 + y^2)}\,. \tag{3.1}$$

Since $x^2 + y^2 > \mathcal{R}^2/4$ we are able to lower bound the distance in Eq. 3.1 and obtain $\left|p_t(j+1) - p_t(j+2)\right|_2 > \mathcal{R}$. This contradicts the assumption that the chain configuration at the beginning of time step t has been communication-valid.

The same reasoning can be applied to determine the position of $p_{t+1}(j)$. Thus, we have proven, that both $p_{t+1}(j)$ and $p_{t+1}(j+1)$ lie within a circle with radius $\mathcal{R}/2$, and so they are in distance at most $\mathcal{R}$. ■

Theorem 3.2. *Assume that in the first time step no station moves by more than* 1 *and that the explorer moves by at most* 1 *per time step. Then the Go-To-The-Middle strategy is movement-valid.*

Proof. We will prove the lemma by induction on the time step. The induction basis holds due to the assumption of the lemma.

Assume that during time step t all relays moved by at most 1. Pick any station v_i. We will show that v_i moves by at most 1 during $t+1$. The position $p_t(i)$ is in the middle of the interval $e := \langle p_{t-1}(i-1), p_{t-1}(i+1)\rangle$. Once again let us lay out a coordinate system, so that e is on the X-axis and $p_t(i)$ is the origin (see Fig. 3.3). Let $a := |e|_2/2$. Obviously it holds $p_{t-1}(i-1) = (-a, 0)$ and $p_{t-1}(i+1) = (a, 0)$. Denote by $(x_{-1}, y_{-1}) = p_t(i-1)$ and $(x_{+1}, y_{+1}) = p_t(i+1)$ the new positions of stations v_{i-1} and v_{i+1}. We want to show that $p_t(i)$ is within distance 1 from $p_{t-1}(i)$.

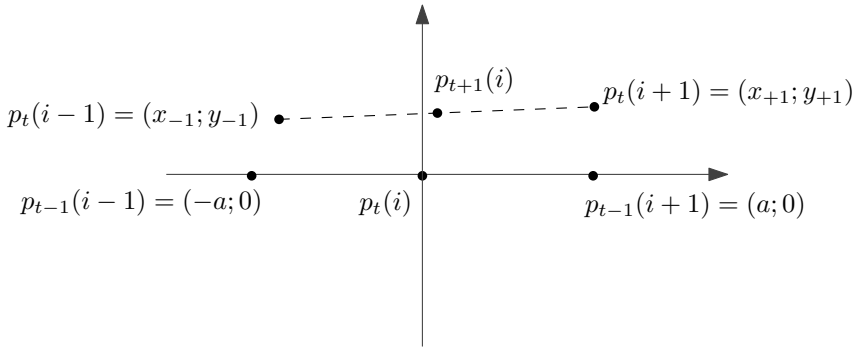


Figure 3.3.: Movement of v_i in $t+1$ depending on movement of v_{i-1} and v_{i+1} in t

Note that by inductive assumption the distances $\left|p_{t-1}(i-1) - p_t(i-1)\right|_2 \leq 1$ and $\left|p_{t-1}(i+1) - p_t(i+1)\right|_2 \leq 1$ and therefore

$$\begin{aligned} (x_{-1} + a)^2 + y_{-1}^2 &\leq 1 \\ (x_{+1} - a)^2 + y_{+1}^2 &\leq 1\,. \end{aligned} \tag{3.2}$$

The coordinates of the point $p_t(i)$ are $\left(\frac{1}{2}(x_{-1}+x_{+1}), \frac{1}{2}(y_{-1}+y_{+1})\right)$. We aim now at showing that

$$\left(\frac{x_{-1}+x_{+1}}{2}\right)^2 + \left(\frac{y_{-1}+y_{+1}}{2}\right)^2 \leq 1\,,$$

which gives us that $p_t(i)$ lies in a circle of radius 1 around $p_{t-1}(i)$. By Eq. (3.2) we obtain

$$\begin{aligned}(x_{-1}+a+x_{+1}-a)^2+(y_{-1}+y_{+1})^2 &\leq (x_{-1}+a)^2+(x_{+1}-a)^2+2\cdot(x_{-1}+a)\cdot(x_{+1}-a)\\ &\quad +y_{-1}^2+y_{+1}^2+2\cdot y_{-1}\cdot y_{+1}\\ &\leq 1+1+2\cdot(x_{-1}+a)\cdot(x_{+1}-a)+2\cdot y_{-1}\cdot y_{+1}\,.\end{aligned}$$

Using the earlier bounds from Eq. (3.2) to express the latter term in terms of y_{-1} and y_{+1} we obtain

$$\begin{aligned}(x_{-1}+a)\cdot(x_{+1}-a)+y_{-1}\cdot y_{+1} &\leq \sqrt{\left(1-y_{-1}^2\right)\cdot\left(1-y_{+1}^2\right)}+y_{-1}\cdot y_{+1}\\ &\leq 1 \qquad (3.3)\end{aligned}$$

A technical proof of Eq. (3.3) can be found in the appendix. Combining the results we have that $p_t(i)$ lies in a circle of radius 1 around $p_{t-1}(i)$. ■

3.2. Potential Function

Before we are able to proceed with the analysis of the *Go-To-The-Middle* strategy in the static scenario, we want to introduce a model which allows us to give a mathematical definition of the progress made by *Go-To-The-Middle* per time step.

For an n-element vector w we will denote by $|w|$ the sum of values of all of its elements, i.e. $|w| = \sum_{i=1}^{n} w[i]$. Similarly $\|w\|$ will denote the sum of the absolute values of all elements, i.e. $\|w\| = \sum_{i=1}^{n} |w[i]|$.

Throughout the analysis of *Go-To-The-Middle*, by L_t we will denote the interval connecting the base camp and the explorer's position $p_t(0)$ from the beginning of time step t. The optimal placement of relays is a uniform distribution on the line L_t. More specifically, for station v_i this is position $p_t^{\text{opt}}(i)$ on the interval L_t in distance $i \cdot d_t/(n_t+1)$ to the explorer. As the distance between neighbored relays is the same for all relays, the fact follows.

Fact 3.3. *Let* $(x_{-1}; y_{-1}) = p_t^{\text{opt}}(i-1)$ *and* $(x_{+1}; y_{+1}) = p_{u,t}^{\text{opt}}(i+1)$. *Then*

$$p_t^{\text{opt}}(i) = \left(\tfrac{1}{2}(x_{-1}+x_{+1}); \tfrac{1}{2}(y_{-1}+y_{+1})\right)\,,$$

for all $i \in [1, n_t]$.

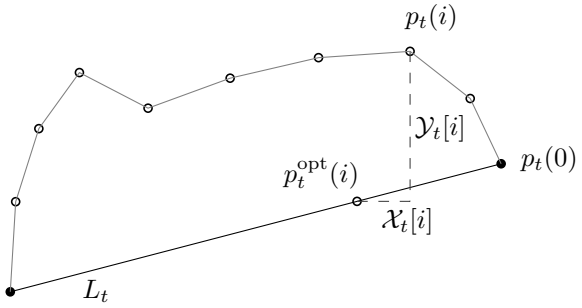


Figure 3.4.: Potential function $\mathcal{X}_t$ and $\mathcal{Y}_t$

We define a family of vectors $\mathcal{X}_t$ and $\mathcal{Y}_t$, such that $\mathcal{X}_t[i]$ denotes the difference in the X-coordinates of $p_t(i)$ and $p_t^{\text{opt}}(i)$ and $\mathcal{Y}_t[i]$ is the same for the Y-coordinates (see Fig. 3.4). More formally, let the coordinates of respectively the current position and the optimal position of station v_i be $(x; y) = p_t(i)$ and $(x^{\text{opt}}; y^{\text{opt}}) = p_t^{\text{opt}}(i)$. Then

$$\mathcal{X}_t[i] = x - x^{\text{opt}}, \;\; \mathcal{Y}_t[i] = y - y^{\text{opt}} .$$

As both the base camp and the explorer lie on the line L_t on their proper positions, we have

$$\mathcal{X}_t[0] = 0, \; \mathcal{Y}_t[0] = 0 \text{ and } \mathcal{X}_t[n_t + 1] = 0, \; \mathcal{Y}_t[n_t + 1] = 0 .$$

Observe, that for two stations v_i and v_{i+1} the condition $\left|p_t(i) - p_t(i+1)\right|_2 \leq \mathcal{R}$ implies $|\mathcal{U}_t[i] - \mathcal{U}_t[i+1]| \leq \mathcal{R}$ yielding the following fact.

Similarly to $\mathcal{X}_t$ and $\mathcal{Y}_t$, by $\mathcal{X}'_t$ and $\mathcal{Y}'_t$ we will denote the vector of distances between the positions of relays after the *Go-To-The-Middle* strategy has been applied and their optimal positions at the beginning of time step t. Formally, let $(x'; y') = p_{t+1}(i)$ and $(x^{\text{opt}}; y^{\text{opt}}) = p_t^{\text{opt}}(i)$. Then

$$\mathcal{X}'_t[i] = x' - x^{\text{opt}}, \;\; \mathcal{Y}'_t[i] = y' - y^{\text{opt}} .$$

for all $i = 1, \ldots, n_t$. For the explorer and base camp we set $\mathcal{X}'_t[0] = 0, \mathcal{Y}'_t[0] = 0$ and $\mathcal{X}'_t[n_t + 1] = 0, \mathcal{Y}'_t[n_t + 1] = 0$. The measure $\mathcal{X}_t$, compared to $\mathcal{X}'_t$, and $\mathcal{Y}_t$ compared to $\mathcal{Y}'_t$, can be used to measure the progress made by the *Go-To-The-Middle* strategy in one time step.

Let $\mathcal{A}$ be a $n_t \times n_t$ matrix with $\mathcal{A}[i, j] = \frac{1}{2}$ for all i, j such that $|i - j| = 1$. For all other i, j

we have $\mathcal{A}[i,j] = 0$:

$$\mathcal{A} = \begin{pmatrix} & 1/2 & & & & & \\ 1/2 & & 1/2 & & & & \\ & 1/2 & & \ddots & & & \\ & & 1/2 & & 1/2 & & \\ & & & \ddots & & 1/2 & \\ & & & & 1/2 & & 1/2 \\ & & & & & 1/2 & \end{pmatrix}$$

The following lemma describes how $\mathcal{X}'_t$ can be computed out of $\mathcal{X}_t$ and $\mathcal{Y}'_t$ from $\mathcal{Y}_t$. By $w[i,\ldots,i+k]$ we will denote a vector of size k composed of the entries $i,\ldots,i+k$ of vector w.

Lemma 3.4. *Assume that the number of relays in the chain does not change in time step t. Then it holds*

$$\mathcal{X}'_t[i] = \begin{cases} 0 & \text{if } i = 0 \\ \frac{1}{2}\left(\mathcal{X}_t[i-1] + \mathcal{X}_t[i+1]\right) & \forall i \in [1,\ldots,n_t] \\ 0 & \text{if } i = n_t + 1\,, \end{cases}$$

and similarly

$$\mathcal{Y}'_t[i] = \begin{cases} 0 & \text{if } i = 0 \\ \frac{1}{2}\left(\mathcal{Y}_t[i-1] + \mathcal{Y}_t[i+1]\right) & \forall i \in [1,\ldots,n_t] \\ 0 & \text{if } i = n_t + 1\,. \end{cases}$$

Equivalently

$$\begin{aligned} \mathcal{X}'_t[1,\ldots,n_t] &= \mathcal{A}\mathcal{X}_t[1,\ldots,n_t] \\ \mathcal{Y}'_t[1,\ldots,n_t] &= \mathcal{A}\mathcal{Y}_t[1,\ldots,n_t]\,. \end{aligned}$$

Proof. Let $(x_{-1}; y_{-1}) = p_t(i-1)$ and $(x_{+1}; y_{+1}) = p_t(i+1)$. Similarly, let $(x^{\text{opt}}_{-1}; y^{\text{opt}}_{-1}) = p^{\text{opt}}_t(i-1)$ and $(x^{\text{opt}}_{+1}; y^{\text{opt}}_{+1}) = p^{\text{opt}}_t(i+1)$. Let v_i be any relay, i.e. $i \in [1, n_t]$. Since *Go-To-The-Middle* moves station v_i exactly into the middle in between $p_t(i-1)$ and $p_t(i+1)$ during time step t, it holds $p_{t+1}(i) = \left(\frac{x_{-1}+x_{+1}}{2}; \frac{x_{-1}+x_{+1}}{2}\right)$. By the definition of $\mathcal{X}_t$ and $\mathcal{Y}_t$ we have

$$\begin{aligned} x_{-1} &= \mathcal{X}_t[i-1] + x^{\text{opt}}_{-1}\,, \quad y_{-1} = \mathcal{Y}_t[i-1] + x^{\text{opt}}_{-1}\,, \\ x_{+1} &= \mathcal{X}_t[i+1] + x^{\text{opt}}_{+1}\,, \quad y_{+1} = \mathcal{Y}_t[i+1] + y^{\text{opt}}_{+1}\,, \end{aligned}$$

Let $(x^{\text{opt}}; y^{\text{opt}}) = p_t^{\text{opt}}(i)$. Then

$$p_{t+1}(i) = \left(\frac{\mathcal{X}_t[i-1] + \mathcal{X}_t[i+1] + x_{-1}^{\text{opt}} + x_{+1}^{\text{opt}}}{2}; \frac{\mathcal{Y}_t[i-1] + \mathcal{Y}_t[i+1] + y_{-1}^{\text{opt}} + y_{+1}^{\text{opt}}}{2} \right)$$

and by Fact 3.3

$$p_{t+1}(i) = \left(\frac{\mathcal{X}_t[i-1] + \mathcal{X}_t[i+1] + 2x}{2}; \frac{\mathcal{Y}_t[i-1] + \mathcal{Y}_t[i+1] + 2y}{2} \right).$$

This proves the lemma for $i \in [1, n_t]$. The boundary conditions hold by definition of $\mathcal{X}'_t$ and $\mathcal{Y}'_t$. ■

3.3. Performance in the Static Scenario

We are now ready to analyze the performance of the *Go-To-The-Middle* strategy for the static scenario, where the explorer does not move. We are given a communication-valid chain at the first time step. We are looking for the number of time steps for the relays to approach their optimal position on the line between the base camp and the explorer.

During the execution of the *Go-To-The-Middle* strategy we do not remove any relays from the path, and as the explorer does not move, no new relays are added. Therefore we can write $n := n_1$.

Note that since the explorer does not move, the line $L := L_t$ is the same for all time steps and therefore $p_t^{\text{opt}}(i)$ is the same for all time steps t. Furthermore the number of relays remains fixed and therefore $\mathcal{X}'_t = \mathcal{X}_{t+1}$ and $\mathcal{Y}'_t = \mathcal{Y}_{t+1}$ and all t.

3.3.1. Lower bound

In the following part we will construct a chain, such that the *Go-To-The-Middle* strategy needs plenty of time to bring the relays near to their positions on the line L. This chain will have all relays on one side of the line L, so that $\mathcal{Y}_t[i] \geq 0$ for all relays v_i if we lay out the coordinate system appropriately. The following lemma shows how the weight of the vector $\mathcal{Y}_t$ decreases when *Go-To-The-Middle* is applied. Bounds on this decrease will allow us to state lower bounds on the performance of *Go-To-The-Middle*.

Lemma 3.5. *Assume that the number of relays in the chain has not changed in time step t. Assume that all stations are on one side of the line L. Then*

$$\|\mathcal{Y}_t\| - \|\mathcal{Y}'_t\| = \frac{|\mathcal{Y}_t[1]| + |\mathcal{Y}_t[n_t]|}{2}.$$

Proof. Since all stations are on the same side of the chain, we have $\left|\mathcal{Y}'_t[i]\right| = \mathcal{Y}'_t[i]$. Then it holds

$$\begin{aligned}\left\|\mathcal{Y}'_t\right\| &= \sum_{i=0}^{n_t+1} \mathcal{Y}'_t[i] = \sum_{i=1}^{n_t} \tfrac{1}{2} \cdot (\mathcal{Y}_t[i-1] + \mathcal{Y}_t[i+1]) \\ &= \sum_{i=0}^{n_t-1} \tfrac{1}{2} \cdot \mathcal{Y}_t[i] + \sum_{i=1}^{n_t} \tfrac{1}{2} \cdot \mathcal{Y}_t[i] \\ &= \tfrac{1}{2} \cdot \mathcal{Y}_t[1] + \sum_{i=1}^{n_t-1} \mathcal{Y}_t[i] + \tfrac{1}{2} \cdot \mathcal{Y}_t[n_t] = \left\|\mathcal{Y}_t\right\| - \left(\tfrac{1}{2} \cdot \mathcal{Y}_t[1] + \tfrac{1}{2} \cdot \mathcal{Y}_t[n_t]\right) .\end{aligned}$$

■

Our lower bound shows that there exist communication-valid configurations, such that the *Go-To-The-Middle* strategy must execute $\Omega(n^2 - \epsilon \cdot n)$ time steps, before each relay is in distance at most ϵ to its optimal position.

Theorem 3.6. *There exists a communication-valid and movement-valid chain of relays such that the Go-To-The-Middle strategy must execute $\Omega(n^2)$ steps in order to obtain $|\mathcal{Y}_t[i]| \leq 1$ for all relays $i = 1, \ldots, n$.*

Proof. We construct the initial chain with $p_1(0) = (0; 0)$, $p_1(n+1) = (n+1; 0)$ and $p_1(i) = (i; i)$ for $i = 1, \ldots, n/2$ and $p_1(i) = (i; n-i)$ for $i = n/2+1, \ldots, n$. Therefore we have $\left\|\mathcal{Y}_1\right\| = n^2/4$ for even n.

By Lemma 3.5 and the fact that the explorer does not move we have

$$\left\|\mathcal{Y}_t\right\| - \left\|\mathcal{Y}_{t+1}\right\| = \tfrac{1}{2}\left(\left|\mathcal{Y}_t[1]\right| + \left|\mathcal{Y}_t[n]\right|\right) .$$

As obviously $\left|\mathcal{Y}_t[j]\right| \leq \mathcal{R}$ for both $j = 1, n$ we have $\left\|\mathcal{Y}_t\right\| + \mathcal{R} \geq \left\|\mathcal{Y}_{t-1}\right\|$. So, after k steps

$$\left\|\mathcal{Y}_k\right\| \geq \left\|\mathcal{Y}_1\right\| - k \cdot \mathcal{R} ,$$

and

$$\left|\mathcal{Y}_k[i]\right| \geq \frac{\left\|\mathcal{Y}_1\right\| - k \cdot \mathcal{R}}{n+2} = \frac{n^2/4 - k \cdot \mathcal{R}}{n+2} .$$

Therefore $\left|\mathcal{Y}_k[i]\right| \leq 1$ only for $k = \Omega(n^2)$. ■

3.3.2. Matrices and Convergence

Before moving on toward the upper bound on the performance of *Go-To-The-Middle* we have to establish some facts about the values of $\mathcal{A}^k$ for large k. Recall that $\mathcal{A}$ is a non-negative $n \times n$ matrix. It possesses the following properties:

- it is sub-stochastic – the sum of its entries in every row and column is at most 1,
- it is symmetric – it holds $\mathcal{A}[i,j] = \mathcal{A}[j,i]$ for every $i, j \in [1,n]$,
- it is irreducible – there exists a k for each pair $i, j \in [1,n]$ such that $\mathcal{A}^k[i,j] > 0$.

In the following lemma we compute the eigenvalues and eigenvectors of the matrix $\mathcal{A}$.

Lemma 3.7. *The eigenvalues of the $n \times n$ matrix $\mathcal{A}$ are*

$$\lambda_j = \cos\left(\frac{\pi \cdot j}{n+1}\right), \quad j = 1, \dots n\ .$$

The corresponding eigenvectors are

$$x_j[i] = \sin\left(\frac{\pi \cdot j \cdot i}{n+1}\right), \quad i = 1, \dots, n, \quad j = 1, \dots, n\ .$$

Proof. We will show that the specified eigenvalues and eigenvectors actually correspond to the matrix $\mathcal{A}$. Thus for each pair of eigenvalue λ_j and eigenvector x_j it must hold

$$\mathcal{A}\, x_j^T = \lambda_j\, x_j^T\ . \tag{3.4}$$

Let us fix some j and prove Eq. (3.4) for this pair λ_j, x_j. The following system of equalities must hold

$$\frac{1}{2}\sin\left(\frac{2j \cdot \pi}{n+1}\right) = \cos\left(\frac{j \cdot \pi}{n+1}\right)\sin\left(\frac{j \cdot \pi}{n+1}\right), \tag{3.5}$$

$$\frac{1}{2}\sin\left(\frac{(i-1) \cdot j \cdot \pi}{n+1}\right) + \frac{1}{2}\sin\left(\frac{(i+1) \cdot j \cdot \pi}{m+1}\right) = \cos\left(\frac{j \cdot \pi}{n+1}\right)\sin\left(\frac{i \cdot j \cdot \pi}{n+1}\right), \tag{3.6}$$

$$\frac{1}{2}\sin\left(\frac{(n-1) \cdot j \cdot \pi}{n+1}\right) = \cos\left(\frac{j \cdot \pi}{n+1}\right)\sin\left(\frac{n \cdot j \cdot \pi}{n+1}\right), \tag{3.7}$$

where Eq. (3.6) must hold for all $i = 2, \dots, n-1$.

We use one of the basic trigonometric identities $\sin(2\alpha) = 2\sin\alpha \cdot \cos\alpha$. With its help Eq. (3.5) follows easily. Similarly we have

$$\sin\alpha + \sin\beta = 2\sin\left(\frac{\alpha+\beta}{2}\right) \cdot \cos\left(\frac{\alpha-\beta}{2}\right),$$

which can be used to prove Eq. (3.6) for each $i = 2, \ldots, n-1$.

For Eq. (3.7) we first note that $\sin(a \cdot j \cdot \pi) = \sin((1-a) \cdot j \cdot \pi)$ for any $j \in \mathbb{N}$ and $0 \leq a < 1$. From this we have

$$\sin\left(\frac{(n-1) \cdot j \cdot \pi}{n+1}\right) = \sin\left(\frac{2j \cdot \pi}{n+1}\right),$$

and

$$\sin\left(\frac{n \cdot j \cdot \pi}{n+1}\right) = \sin\left(\frac{j \cdot \pi}{n+1}\right).$$

This reduces Eq. (3.7) to Eq. (3.6). ■

The following is a widely known fact following from the spectral theorem.

Theorem 3.8. *Let P be a symmetric $n \times n$ matrix with eigenvalues $\lambda_1, \ldots, \lambda_n$ and eigenvectors $x_1, \ldots, x_n$. Then we can write $P = NDN^{-1}$ where $N = \left[x_1^T, \ldots, x_n^T\right]$ and $D[i,i] = \lambda_i$, $D[i,j] = 0$ for $i \neq j$.*

The following lemma is similar to a convergence result by Cinlar [Cin75]. We rewrite it for sub-stochastic matrices here.

Lemma 3.9. *For any irreducible, symmetric, substochastic $n \times n$ matrix P with pairwise distinct eigenvalues and any i, j we have*

$$P^k[i,j] \leq n \cdot \alpha \cdot \beta^k,$$

where β is the largest absolute value of eigenvalues of the matrix P and $\alpha = \max_{i,j,i',j'} |x_j[i] \cdot x_{j'}[i']|$ with x_j denoting the j-th eigenvector of matrix P.

Proof. Let $\lambda_1, \ldots, \lambda_n$ denote the eigenvalues of matrix P, ordered descending by their absolute value. As P has pairwise distinct eigenvalues, we can write by Fact 3.8 $P = NDN^{-1}$, $N = \left[x_1^T, \ldots, x_n^T\right]$, and $D[i,i] = \lambda_i$, $D[i,j] = 0$ for $i \neq j$. Denoting rows of N^{-1} as vectors $\pi_1, \ldots, \pi_n$ we can define the matrices $B_k = x_k^T \cdot \pi_k$,

$$B_k = \begin{bmatrix} x_k[1] \\ \vdots \\ x_k[n] \end{bmatrix} [\pi_k[1], \ldots, \pi_k[n]] = \begin{bmatrix} x_k[1] \cdot \pi_k[1] & \ldots & x_k[1] \cdot \pi_k[n] \\ \vdots & & \vdots \\ x_k[n] \cdot \pi_k[1] & \ldots & x_k[n] \cdot \pi_k[n] \end{bmatrix}.$$

We can write P as

$$P = \lambda_1 B_1 + \cdots + \lambda_n B_n.$$

Observe that $N^{-1} N = I$ implies $\pi_j \cdot x_k = 0$ for $j \neq k$ and $\pi_j \cdot x_j = 1$. Thus $B_j B_k$ is equal to 0 when $j \neq k$ and B_k when $j = k$. The expansion of $(\lambda_1 B_1 + \cdots + \lambda_n B_n)^k$ obtained by applying the binomial theorem thereby reduces to

$$P^k = \lambda_1^k B_1 + \lambda_2^k B_2 + \ldots + \lambda_n^k B_n \, .$$

Now we can write

$$\begin{aligned} |P^k[i,j]| &= |\lambda_1^k B_1[i,j] + \lambda_2^k B_2[i,j] + \ldots + \lambda_n^k B_n[i,j]| \, , \\ &\leq |\lambda_1^k| \left| B_1[i,j] + \ldots + B_n[i,j] \right| \, , \\ &\leq |\lambda_1^k| \left(\left| B_1[i,j] \right| + \ldots + \left| B_n[i,j] \right| \right) \, . \end{aligned}$$

Each of the terms $|B_k[i,j]|$ can be bounded by $\max_{i,j,i',j'} |x_j[i] \cdot \pi_{j'}[i']|$. As the matrix P is symmetric, its left-hand and right-hand eigenvectors are essentially the same (they are only transposed). Thus we can rewrite the former term to $\max_{i,j,i',j'} |x_j[i] \cdot x'_j(i')|$.

Finally, since the matrix P is non-negative, all its powers are also non-negative and we can write $|P^k[i,j]| = P^k[i,j]$. ■

3.3.3. Upper bound

We present an upper bound showing that the *Go-To-The-Middle* strategy needs at most $O(n^2 \log(n/\epsilon))$ time steps for each station to be in distance at most ϵ to its optimal position in both the X and Y-coordinate.

Theorem 3.10. *For any communication-valid chain with n relays, after* $t \geq 9 \cdot (n+1)^2 \cdot \log((n+1)/\epsilon)$ *time steps of execution of Go-To-The-Middle it holds* $|\mathcal{X}_t[i]| \leq \varepsilon$ *and* $\left|\mathcal{Y}_t[i]\right| \leq \varepsilon$ *for any* $\varepsilon > 0$, *any* $i \in [1,n]$.

Proof. We will show the theorem for $\left|\mathcal{Y}_t[i]\right| \leq \varepsilon$, while the proof for $\mathcal{X}$ is analogous. For ease of presentation we define $z_t = \mathcal{Y}_t[1,\ldots,n]$. Therefore z_t is the vector of distances of all relays (excluding the base camp and the explorer) to their optimal positions.

Let β be the largest absolute value of the eigenvalues of $\mathcal{A}$ and $\beta = \max_{i,j,i',j'} \left| x_j[i] \cdot x_{j'}[i] \right|$ where x_j are the eigenvectors of $\mathcal{A}$. As all entries of all eigenvectors of $\mathcal{A}$ are not larger than 1, we have $\beta \leq 1$. Observing the eigenvalues of $\mathcal{A}$ we can quickly determine that $\beta = \left|\cos \frac{\pi}{n+1}\right|$. We want to have an upper bound on β which has a cleaner form then this. Thus, let us expand $\cos x$ around $x = 0$ from the Taylor series. We obtain

$$\cos x = 1 - \frac{x^2}{2} + \frac{x^4}{24} - \frac{\sin(y) \cdot x^5}{5!} \, ,$$

where $0 < y < x$. Setting $x = \pi/(n+1)$ for sufficiently large n we have $\sin(y) > 0$ and therefore

$$\begin{aligned}\cos\left(\frac{\pi}{n+1}\right) &\leq 1 - \frac{\pi^2}{2(n+1)^2} + \frac{\pi^4}{24(n+1)^4} \\ &\leq 1 - \frac{1}{(n+1)^2} + 5\frac{1}{(n+1)^4}\,.\end{aligned}$$

Since $5/(n+1)^4 \leq 1/2(n+1)^2$ for sufficiently large n we have

$$\cos\frac{\pi}{n+1} \leq 1 - \frac{1}{2(n+1)^2}\,.$$

This lets us conclude that for $t = 2(n+1)^2$ we obtain $\beta^t \leq 1/e$ and that for $\tau = 2(n+1)^2 \cdot \ln\left(\frac{\mathcal{R}\cdot n^3}{\varepsilon}\right)$ it holds $\beta^\tau \leq \frac{\varepsilon}{\mathcal{R}\cdot n^3}$. Therefore for all i, j

$$\mathcal{A}^\tau[i,j] \leq \frac{\varepsilon}{\mathcal{R}\cdot n^2}\,.$$

Recall that by Lemma 3.4

$$z_t = \mathcal{A}^{t-1}\cdot z_1\,.$$

Since the distance between nodes is at most $\mathcal{R}$, we know that for any relay v_i it holds $z_t[i] \leq \mathcal{R}\cdot n$. On the other hand we know that entries of A^t are always non-negative. So

$$z_\tau[i] = z_1\cdot\mathcal{A}^\tau[i] \leq n^2\cdot\mathcal{R}\cdot\frac{\varepsilon}{\mathcal{R}\cdot n^2} \leq \varepsilon\,.$$

■

3.4. Performance in the Dynamic Scenario

In the following part we will analyze the performance of the *Go-To-The-Middle* strategy in the dynamic scenario. Here we assume that at the beginning of the first time step stations are located on their optimal positions on the line to the explorer. We then let an adversary control the behavior of the explorer and look at the performance of the *Go-To-The-Middle* strategy.

Recall that $\mathcal{X}'_t$ and $\mathcal{Y}'_t$ is the distance between the positions of relays at the end of time step t (i.e. $p_{t+1}(i)$) and the line L_t, while $\mathcal{X}_{t+1}$ and $\mathcal{Y}_{t+1}$ is the distance to line L_{t+1}. Then, the difference between $\mathcal{X}'_t, \mathcal{Y}'_t$ and $\mathcal{X}_{t+1}, \mathcal{Y}_{t+1}$ is only dependent on the movement of the explorer, as depicted on Fig 3.5.

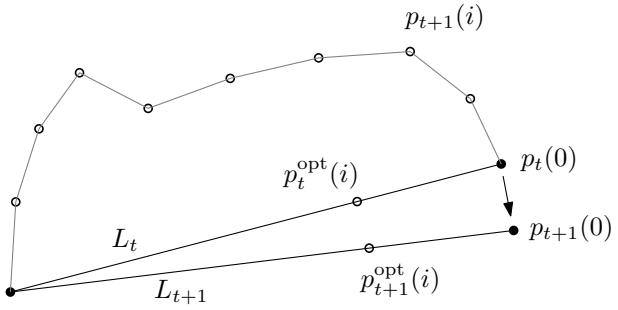


Figure 3.5.: Movement of explorer

Number of Relays

When considering a dynamic scenario with a moving explorer we have to deal with the issue that the number of relays in the chain may have to be changed.

Although the task of inserting and removing relays to the chain should intuitively be the responsibility of the base camp, we assign the responsibility for controlling the number of relays in the chain to the explorer. This is the only possibility, as with the *Go-To-The-Middle* strategy the base camp cannot know of movement of the explorer and the changed requirements on the number of relays in the chain. The explorer is thereby maintaining the invariant $n_t = \gamma \cdot d_t/\mathcal{R}$ by inserting new relays into the chain whenever it increases its distance to the explorer and removing them in the opposite case. The constant $\gamma > 1$ introduces a slackness by including more relays into the chain than would be necessary. A slightly larger number of relays is necessary for the chain to work properly. The new relays are inserted as first in the chain, causing a renumbering of the whole chain. Similarly, it is always the first relay v_1 which is removed from the chain. Note, that this technique requires the explorer to know d_t in order to control the number of relays. This may be considered a large drawback of the *Go-To-The-Middle* strategy.

The fact that relays are inserted at the explorer's end of the chain requires the explorer to carry a sufficiently large pool of relays. If this is not feasible and spare relays are only available at the base camp, a different schema may be used. We assign each relay v_i a partner, which is following the movement of v_i. During each *Go-To-The-Middle* step, a relay and its partner perform the same movement. Afterward, the partner of v_i moves to v_{i-1} and becomes its partner. At the base camp, a new relay is introduced in each time step as the partner of v_n. At the explorer there are two possibilities. If the explorer has not moved far away from v_1, the partner of v_n starts going back to the base camp and is reused there. If a new relay is needed to hold up the communication with the explorer, the partner of v_n is used. This allows the explorer to have relays available whenever

needed. On the other hand, it doubles the number of relays used and imposes a large energy cost due to the constant movement of spare relays along the chain.

Intuition

Before providing formal lower and upper bounds on the performance of the *Go-To-The-Middle* strategy in the dynamic setting, we want to give an example which shows how the *Go-To-The-Middle* strategy can perform in a dynamic environment. This example does not fit directly into our base camp-explorer model, while it provides a simplified model we can handle more easily.

Assume that the chain of relays is spanned between two mobile explorers. Both of these explorers will be moving on parallel lines in the same direction. Fix a coordinate system, such that the explorers are only decreasing their Y-coordinate while moving (i.e. they are moving upwards in the coordinate system all the time). Therefore we can utilize the measure $\mathcal{Y}_t$ to describe the distance of the stations to the horizontal line L_t spanned between the two explorers.

We start with n relays forming the chain between the two explorers and assume that no relays are removed or added to the chain. As the distance between the two explorers remains constant, this is reasonable. We want to investigate whether there exists some stability point, such that the explorers move by some distance e, increasing the distance of relays to L_t, while at the same time this distance is decreased by the *Go-To-The-Middle* strategy to the value prior to the movement of the explorer. In other words, we are looking for such positions of the relays and such a movement distance e, that the movement of the explorer is canceled out by the *Go-To-The-Middle* strategy in each time step.

We assume that it holds $\mathcal{X}_1 = 0$. Since the only movement of the explorer is in vertical direction, the relays do not exhibit any horizontal movement, therefore $\mathcal{X}_t = 0$ for all time steps. As the explorer moves by e it holds $\mathcal{Y}'_t[i] = \mathcal{Y}_{t+1}[i] - e$ for all $1 \leq i \leq n$.

For the sake of compactness of notation describe by $z(i) := \mathcal{Y}_t[i]$ and $z'(i) := \mathcal{Y}'_t[i]$. Note, that we are treating z as a discrete function defined on $[0, n+1]$ rather than a vector. To have our balance property fulfilled we are looking for a function z defining the positions of relays in such a way that *Go-To-The-Middle* decreases the distance of relays to L_t in one time step be e, i.e. $z(i) = z'(i) + e$ for all $1 \leq i \leq n$. We restrict our solution space to polynomials of second-degree, so that $z(i) = a \cdot i^2 + b \cdot i + c$. Summarizing

our balance properties it must hold

$$\begin{aligned} z(0) &= 0\,, \\ z(n+1) &= 0\,, \\ z'(i) &= \tfrac{1}{2}\cdot(z(i-1)+z(i+1))\,, \\ z'(i) &= z(i)-e\,. \end{aligned}$$

Using these equalities we obtain $a = -e$ and $b = -a\cdot(n+1)$ for our polynomial. Therefore a chain characterized by

$$z(i) = -e\cdot i^2 + e\cdot(n+1)\cdot i\,, \tag{3.8}$$

will fulfill our balance property when the explorer moves by e.

We now add another important restriction on the values of $z(i)$: since the relays have to stay in communication distance, we should have $z(i) - z(i-1) \leq \mathcal{R}$ and in particular $z(1) - z(0) \leq \mathcal{R}$. Plugging our solution to $z(i)$ from Eq. (3.8) into the latter restriction we obtain $e \leq \mathcal{R}/n$. Therefore if $z(i)$ is given by a second-degree polynomial the greatest speed for the explorer which does not violate any of the necessary properties is $\mathcal{R}/n$. This results yields the intuition that the speed $\mathcal{R}/n$ is an important threshold for the *Go-To-The-Middle* strategy. Obviously this does not constitute a formal lower bound or upper bound. These will follow in the next sections.

3.4.1. Upper bound

We present an upper bound on the performance of the *Go-To-The-Middle* strategy, by showing that on average the explorer will not be hindered to a speed lower than $\Omega(1/n)$. Recall that we are only dealing with movement of the explorer in the same distance to the base camp, therefore $d = d_t$ and $n = n_t$. We will show the following theorem.

Theorem 3.11. *Let the explorer move on a circle with diameter d around the base camp. Then its average speed is* $\Omega(1/d)$.

For the purpose of proving Theorem 3.11 we have to show that the *Go-To-The-Middle* strategy decreases the value of the vectors $\mathcal{X}_t$ and $\mathcal{Y}_t$ at least by some constant factor in every time step.

Lemma 3.12. *Assume that the number of relays in the chain has not changed in time step t. Then it holds*

$$\|\mathcal{X}_t\| - \left\|\mathcal{X}'_t\right\| \geq \frac{|\mathcal{X}_t[1]| + |\mathcal{X}_t[n]|}{2}\,,$$

and

$$\left\|\mathcal{Y}_t\right\| - \left\|\mathcal{Y}'_t\right\| \geq \frac{\left|\mathcal{Y}_t[1]\right| + \left|\mathcal{Y}_t[n]\right|}{2} .$$

Proof. The statement of this lemma is similar to that from Lemma 3.5. The difference is that here we do not assume the relays to lie on one side of the line L_t. The following estimate relies on the property expressed in Lemma 3.4

$$\begin{aligned}
\left\|\mathcal{Y}'_t\right\| &= \sum_{i=0}^{n} \left|\mathcal{Y}'_t[i]\right| = \sum_{i=1}^{n} \tfrac{1}{2} \left|\mathcal{Y}_t[i-1] + \mathcal{Y}_t[i+1]\right| \\
&\leq \sum_{i=0}^{n-1} \tfrac{1}{2} \cdot \left|\mathcal{Y}_t[i]\right| + \sum_{i=1}^{n} \tfrac{1}{2} \cdot \left|\mathcal{Y}_t[i]\right| \\
&= \tfrac{1}{2} \cdot \left|\mathcal{Y}_t[1]\right| + \sum_{i=2}^{n-1} \left|\mathcal{Y}_t[i]\right| + \tfrac{1}{2} \cdot \left|\mathcal{Y}_t[n]\right| \\
&= \left\|\mathcal{Y}_t\right\| - \tfrac{1}{2} \cdot \left(\left|\mathcal{Y}_t[1]\right| + \left|\mathcal{Y}_t[n]\right|\right) .
\end{aligned}$$

The bound works in the same way for $\mathcal{X}'_t$. ■

Complementary to the last statement we want to show that the increase of the vectors $\mathcal{X}_t, \mathcal{Y}_t$ caused by the explorer's movement is bounded from above.

Lemma 3.13. *Let the length of the explorer's movement vector be e in time step $t-1$. Then*

$$\|\mathcal{X}_t\| - \|\mathcal{X}'_{t-1}\| \leq e \cdot n ,$$

and

$$\|\mathcal{Y}_t\| - \|\mathcal{Y}'_{t-1}\| \leq e \cdot n .$$

Proof. Recall that the optimal position $p_t^{\text{opt}}(i)$ of the relay v_i is located on the interval L_t in distance $i \cdot d/(n+1)$ to the base camp. By assumption we have $\left|p_t(0) - p_{t+1}(0)\right|_2 = e$ and obviously $p_t(0)$ is located on L_t and $p_{t+1}(0)$ on L_{t+1}. Take now the point $p_t^{\text{opt}}(i)$ on the interval L_t and the point $p_{t+1}^{\text{opt}}(i)$ located on the interval L_{t+1}. By a geometric argument, we have $\left|p_t^{\text{opt}}(i) - p_{t+1}^{\text{opt}}(i)\right|_2 \leq e$ for all $i \in [1,n]$. Let $p_t^{\text{opt}}(i) = (x;y)$ and $p_{t+1}^{\text{opt}}(i) = (x';y')$. The bound on the euclidean distance between $p_t^{\text{opt}}(i)$ and $p_{t+1}^{\text{opt}}(i)$ implies $|x - x'| \leq e$ and $|y - y'| \leq q$. From this we can follow that $|\mathcal{X}_t[i]| - \left|\mathcal{X}'_{t-1}[i]\right| \leq e$ and similarly $\left|\mathcal{Y}_t[i]\right| - \left|\mathcal{Y}'_{t-1}[i]\right| \leq e$. Summing up over all relay we obtain the statement of the lemma. ■

Take any movement sequence of the explorer on a circle with diameter d around the base camp. The sequence can be described by a sequence of lengths of the movement vectors of the explorer $e_1, \ldots, e_m$. We only care for the lengths of the vectors here, neglecting their direction. We assume that in a step t the explorer either moves with its maximum speed 1 (i.e. $e_t = 1$), or that its movement is restricted by the relay chain and $e_t < 1$. Denote by T the set of all time steps t such that $e_t < 1$. For each time step out of T the following important fact holds.

Lemma 3.14. *There exists a constant $\varphi > 0$ such that for each time step $t \in T$ it holds*

$$|\mathcal{X}_t[1]| + \left|\mathcal{Y}_t[1]\right| \geq \varphi$$

Proof. If the explorer is restricted in time step t, it means that $\left|p_t(1) - p_t(0)\right|_2 > \mathcal{R} - 2$. Otherwise the explorer could move for a distance of 1 without exceeding the distance $\mathcal{R}$ to v_1 at the end of time step t, no matter how v_1 moved during t.

As the number of relays in the chain is given by $n = \gamma \cdot d/\mathcal{R}$, we have

$$|p_t^{\text{opt}}(1) - p_t(0)|_2 \quad = \quad \frac{d}{n+1} \leq d/n = \mathcal{R}/\gamma \,.$$

By the triangle inequality

$$\left|p_t(1) - p_t(0)\right|_2 \leq \left|p_t(1) - p_t^{\text{opt}}(1)\right|_2 + \left|p_t^{\text{opt}}(1) - p_t(0)\right|_2 \,,$$

and therefore

$$\left|p_t(1) - p_t^{\text{opt}}(1)\right|_2 > \mathcal{R} - 2 - \mathcal{R}/\gamma \,,$$

and therefore

$$\left|\mathcal{Y}_t[1]\right| + \left|\mathcal{Y}_t[1]\right| > \mathcal{R} - 2 - \mathcal{R}/\gamma \,.$$

Setting $\varphi = \mathcal{R} - 2 - \mathcal{R}/\gamma$ which is greater than zero for sufficiently large $\mathcal{R}$ and γ we can follow the statement of the lemma. ■

We can now move on to proving the upper bound on the performance of *Go-To-The-Middle*.

Proof of Theorem 3.11. The average speed of the explorer during the whole execution is given by $\left(\sum_{i=1}^{m} e_i\right)/m$. We want to bound the average speed from below. By Lemma 3.13 we have

$$\begin{aligned} \|\mathcal{X}_{t+1}\| - \left\|\mathcal{X}'_t\right\| &\leq e_t \cdot n \,, \\ \left\|\mathcal{Y}_{t+1}\right\| - \left\|\mathcal{Y}'_t\right\| &\leq e_t \cdot n \,. \end{aligned}$$

Furthermore by Lemma 3.12 and Lemma 3.14 for all $t \in T$ holds

$$\|\mathcal{X}_t\| - \left\|\mathcal{X}'_t\right\| + \left\|\mathcal{Y}_t\right\| - \left\|\mathcal{Y}'_t\right\| \geq \frac{|\mathcal{X}_t[1]| + |\mathcal{X}_t[n]|}{2} + \frac{\left|\mathcal{Y}_t[1]\right| + \left|\mathcal{Y}_t[n]\right|}{2} \geq \varphi/2 \,.$$

Let us now introduce a potential function Φ. At the beginning we have $\Phi_1 = \|\mathcal{X}_1\| + \left\|\mathcal{Y}_1\right\|$. We define recursively

$$\begin{aligned} \Phi_{t+1} &= \Phi_t + \|\mathcal{X}_{t+1}\| - \left\|\mathcal{X}'_t\right\| - \left(\|\mathcal{X}_t\| - \left\|\mathcal{X}'_t\right\|\right) + \left\|\mathcal{Y}_{t+1}\right\| - \left\|\mathcal{Y}'_t\right\| - \left(\left\|\mathcal{Y}_t\right\| - \left\|\mathcal{Y}'_t\right\|\right) \\ &= \|\mathcal{X}_{t+1}\| + \left\|\mathcal{Y}_{t+1}\right\| \,. \end{aligned}$$

Therefore we have

$$\begin{aligned} \Phi_m &= \Phi_1 + \sum_{i=1}^{m-1} \left(\|\mathcal{X}_{i+1}\| - \left\|\mathcal{X}'_i\right\| + \left|\mathcal{Y}_{i+1}\right\| - \left\|\mathcal{Y}'_i\right\|\right) \\ &\quad - \sum_{i=1}^{m-1} \left(\|\mathcal{X}_i\| - \left\|\mathcal{X}'_i\right\| + \left\|\mathcal{Y}_i\right\| - \left\|\mathcal{Y}'_i\right\|\right) \\ &\leq \Phi_1 + 2n \cdot \sum_{i=1}^{m-1} e_i - |T| \cdot \varphi/2 \,. \end{aligned}$$

The starting condition for our upper bound was that all relays are placed on their optimal positions in time step 1, so that $\Phi_1 = 0$. As $\Phi_t = \|\mathcal{X}_{t+1}| + \left\|\mathcal{Y}_{t+1}\right\| \geq 0$ it follows

$$\sum_{i=1}^{m} e_i \geq \frac{|T| \cdot \varphi}{4n} \,.$$

Since the explorer has moved by 1 in all time steps not included in T we have another bound $\sum_{i=1}^{m} e_i \geq m - |T|$. Combining both bounds we obtain

$$\frac{\sum_{i=1}^{m} e_i}{m} \geq \frac{|T| \cdot \varphi}{8n \cdot m} + \frac{m - |T|}{2m} \,.$$

Evaluating the two possibilities $|T| \geq m/2$ and $|T| < m/2$ we obtain

$$\frac{\sum_{i=1}^{m} e_i}{m} \geq \frac{1}{16n} \,.$$

■

3.4.2. Lower bound

For the lower bound on the performance of the *Go-To-The-Middle* strategy we will show the following theorem.

Theorem 3.15. *There exists a movement pattern for the explorer such that the Go-To-The-Middle strategy forces the explorer to slow down to a speed of $O(1/d)$.*

For the proof of the lower bound we will introduce a measure $\mathcal{V}_t$. We will show in Lemma 3.19 that a movement of the explorer by some vector e around the base camp increases the value of $\mathcal{V}_t$ by at least $\Omega(n \cdot e)$. On the other hand, in Lemma 3.20 we will show that the *Go-To-The-Middle* strategy is able to decrease the value of $\mathcal{V}_t$ by at most some constant factor during one time step. The measure $\mathcal{V}_t$ will be defined in such a way, that its value over some specific threshold implies that the relay chain got disconnected. As the *Go-To-The-Middle* strategy and a properly behaving explorer guarantee that this does not happen, both the increase and decrease of the measure $\mathcal{V}_t$ must balance out in the long run. Therefore, the explorer will be forced to slow down to an average of $O(1/d)$.

For the construction of the lower bound we will be moving the explorer in one direction around the base camp. Let us fix this direction to clockwise. Let $L_{B,t}$ be a line orthogonal to L_t, starting at the base camp. The line $L_{B,t}$ divides the plane in two half-planes. The half-plane occupied by the explorer will be denoted F_t. The following technical lemma shows that the relays in the beginning of the chain are always located in F_t.

Lemma 3.16. *The first $\lfloor d/\mathcal{R} \rfloor$ relays are in F_t.*

Proof. Assume that a relay v_i with $i \leq \lfloor d/\mathcal{R} \rfloor$ is on the other side of the line $L_{B,t}$ than the explorer. Therefore $\lfloor d/\mathcal{R} \rfloor$ relays are used to construct a chain of length at least d. This implies that the distance between at least two relays within the chain is more than $\mathcal{R}$, leading to a contradiction. ■

For the sake of simplicity of notation let the set S denote the first $\lfloor d/\mathcal{R} \rfloor$ relays in the chain. By $\mathcal{V}_t[i]$ we denote the distance between $p_t(i)$ and the line L_t (understood in the usual way as the length of the shortest line segment between $p_t(i)$ and L_t). The measure $\mathcal{V}$ is only defined for relays in S. Let the distance be positive if the station is on the counter-clockwise side of L_t and negative otherwise. Since all stations are in F_t, the directions are unambiguous. We define $\mathcal{V}'_t$ as the distance between $p_{t+1}(i)$ and L_t for all $i = 1, \ldots, n$ and $\mathcal{V}'_t[0] = 0$, $\mathcal{V}'_t[n+1] = 0$. Define by $\tilde{p}_t(i)$ the projection of the position $p_t(i)$ on the line L_{t-1}.

Observe, that for two relays v_i and v_{i+1} the condition $\left|p_t(i) - p_t(i+1)\right|_2 \leq \mathcal{R}$ implies $|\mathcal{V}_t[i] - \mathcal{V}_t[i+1]| \leq \mathcal{R}$. For relay v_i in the chain it must hold $\mathcal{V}_t[i] \leq i \cdot \mathcal{R}$ as otherwise the chain would be disconnected. From this, the following fact follows.

Fact 3.17. *In order for the chain part included in S to keep connected it must hold*

$$|\mathcal{V}_t| \leq |S| \cdot n \cdot \mathcal{R} \leq n^2 \cdot \mathcal{R} \,. \tag{3.9}$$

The following lemma shows the easy fact that at least half of the relays in S are in distance at least $d/2$ to the base camp.

Lemma 3.18. *There are $|S|/2$ relays in S, such that for each of them $\left|b - p_t(i)\right|_2 \geq d/2$ and $\left|b - \tilde{p}_t(i)\right|_2 \geq d/2$.*

Proof. First observe that for any relay v_i it holds

$$\left|b - \tilde{p}_t(i)\right|_2 \leq \left|b - p_t(i)\right|_2 \,. \tag{3.10}$$

It is clear that relay v_i can be in distance at most $i \cdot \mathcal{R}$ from the explorer, otherwise the chain between v_i and v_0 got disconnected. Formally $\left|p_t(i) - p_t(0)\right|_2 \leq i \cdot \mathcal{R}$. Therefore, $\left|p_t(i) - b\right|_2 \geq d - i \cdot \mathcal{R}$. For all station with indices not greater than $d/2\mathcal{R}$ it holds $\left|p_t(i) - b\right|_2 \geq d - i \cdot \mathcal{R} \geq d/2$. Together with Eq. (3.10) this yields the claim. ■

In the following lemma we investigate how a movement of the explorer influences the measure $\mathcal{V}$.

Lemma 3.19. *Let the length of the explorer's movement vector be e in time step $t-1$. Then*

$$|\mathcal{V}_t| - \left|\mathcal{V}'_{t-1}\right| \geq \Omega(e \cdot n)$$

Proof. Let us choose any relay v_i from set S. Denote by $\tilde{r}$ the distance between the base camp and the projection of $p_t(i)$ on L_{t-1}, i.e. $\tilde{r} = |b - \tilde{p}_t(i)|_2$ and similarly $r = |b - p_t(i)|_2$. We will show that $\mathcal{V}_t[i] - \mathcal{V}'_{t-1}[i] \geq \Omega(e)$, and for that purpose we have to distinguish four cases, depending on the sign of $\mathcal{V}'_{t-1}[i]$ and $\mathcal{V}_t[i]$.

Case $\mathcal{V}'_{t-1} \geq 0$ and $\mathcal{V}_t \geq 0$. For simplicity of notation let $z_{t-1} = \mathcal{V}'_{t-1}[i]$ and $z_t = \mathcal{V}_t[i]$. Let both angles α and β be rooted at b. Angle α is the angle between L_{t-1} and L_t. Angle β is spanned between the interval $\langle b, p_t(i)\rangle$ and L_{t-1}. From obvious geometrical observations

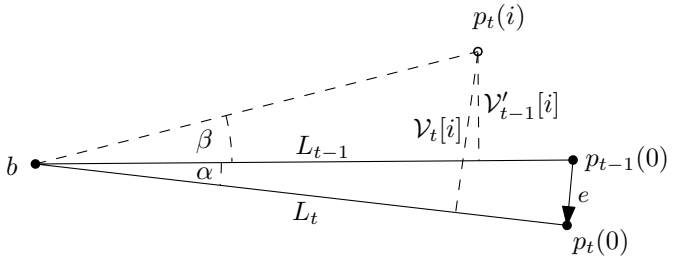


Figure 3.6.: Case $\mathcal{V}'_{t-1} \geq 0$ and $\mathcal{V}_t \geq 0$

we can determine the cosine and sine of both angles, so that

$$\begin{aligned} \cos\alpha &= 1 - \frac{e^2}{2d^2}, \quad \sin\alpha = \sqrt{\frac{e^2}{d^2} - \frac{e^4}{4d^4}} \\ \cos\beta &= \frac{\tilde{r}}{\sqrt{z_{t-1}^2 + \tilde{r}^2}}, \quad \sin\beta = \frac{z_{t-1}}{\sqrt{z_{t-1}^2 + \tilde{r}^2}}. \end{aligned}$$

Therefore

$$\begin{aligned} \sin(\alpha+\beta) &= \sin\alpha\cos\beta + \cos\alpha\sin\beta \\ &= \frac{e}{d} \cdot \sqrt{1 - \frac{e^2}{4d^2}} \cdot \frac{\tilde{r}}{\sqrt{z_{t-1}^2 + \tilde{r}^2}} + \\ &\quad \left(1 - \frac{e^2}{2d^2}\right) \cdot \frac{z_{t-1}}{\sqrt{z_{t-1}^2 + \tilde{r}^2}}. \end{aligned}$$

In order to bound $z_t - z_{t-1}$ from below we use the fact that

$$sin(\alpha+\beta) = \frac{z_t}{\sqrt{z_{t-1}^2 + \tilde{r}^2}},$$

and hereby

$$\begin{aligned} z_t &= \sqrt{z_{t-1}^2 + \tilde{r}^2} \cdot \sin(\alpha+\beta) \\ &= z_{t-1} + \frac{\tilde{r} \cdot e}{d} \cdot \sqrt{1 - \frac{e^2}{4d^2}} - z_{t-1} \cdot \frac{e^2}{2d^2}. \end{aligned}$$

We can bound $z_{t-1} \leq n \cdot \mathcal{R}$ and then

$$z_{t-1} \cdot \frac{e^2}{2d^2} \leq \frac{\gamma^2 \cdot e^2}{2n} \leq O(e/n),$$

since $d = \mathcal{R} \cdot n/\gamma$ and $e \leq 1$. For sufficiently large d we have

$$\sqrt{1 - \frac{e^2}{4d^2}} \geq 1/2 \,.$$

Therefore

$$\mathcal{V}_t[i] - \mathcal{V}'_{t-1}[i] = z_t - z_{t-1} \quad \geq \quad \frac{\tilde{r} \cdot e}{2d} - O(e/n) \,. \tag{3.11}$$

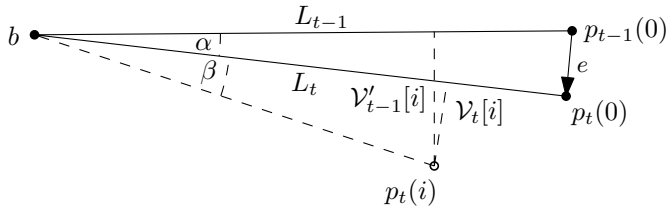


Figure 3.7.: Case $\mathcal{V}'_{t-1}[i] \leq 0$ and $\mathcal{V}_t[i] \leq 0$

Case $\mathcal{V}'_{t-1}[i] \leq 0$ and $\mathcal{V}_t[i] \leq 0$. Here we are dealing with a situation very similar to the former case. Laying $z_t = -\mathcal{V}'_{t-1}[i]$ and $z_{t-1} = -\mathcal{V}_t[i]$ we can use Eq. (3.11) and obtain

$$\mathcal{V}_t[i] - \mathcal{V}'_{t-1}[i] = -z_{t-1} + z_t \geq \frac{\tilde{r} \cdot e}{2d} - O(e/n) \,.$$

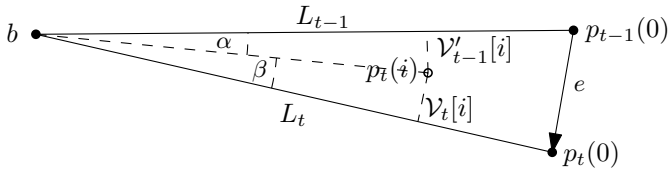


Figure 3.8.: Case $\mathcal{V}'_{t-1}[i] \leq 0$ and $\mathcal{V}_t[i] \geq 0$

Case $\mathcal{V}'_{t-1}[i] \leq 0$ and $\mathcal{V}_t[i] \geq 0$. In this case, $p_t(i)$ lies "in between" the intervals L_t and L_{t-1} as shown on Fig. 3.8. We will show that either $\max\{-\mathcal{V}'_{t-1}[i], \mathcal{V}_t[i]\} \geq r \cdot e/2d$. This implies

$$\mathcal{V}_t[i] - \mathcal{V}'_{t-1}[i] \geq \frac{r \cdot e}{2d} \,.$$

Let us assume without loss of generality that $\mathcal{V}_t[i] \geq -\mathcal{V}'_{t-1}[i]$. For simplicity of notation let $z := \mathcal{V}_t[i]$. Let α be the angle between $\langle b, p_t(i)\rangle$ and L_{t-1} and β between $\langle b, p_t(i)\rangle$ and L_t. As $\mathcal{V}_t[i] \geq -\mathcal{V}'_{t-1}[i]$ we have $\beta \geq \frac{1}{2}(\alpha+\beta)$. As e is reasonably small in comparison to d, we have $\beta \leq \pi/2$ and therefore $z = r \cdot \sin\beta \geq r \cdot \sin\frac{1}{2}(\alpha+\beta)$. By one of the common trigonometrical identities we have

$$z \geq r \cdot \sin\tfrac{1}{2}(\alpha+\beta) \geq \sqrt{\frac{1-\cos(\alpha+\beta)}{2}}\,.$$

From the Law of Cosines it holds $\cos(\alpha+\beta) = 1 - e^2/2d^2$, so that eventually $z \geq r \cdot e/2d$.

Case $\mathcal{V}'_{t-1}[i] \geq 0$ and $\mathcal{V}_t[i] \leq 0$. This is impossible to happen, since the explorer moves in clockwise direction.

Bringing all cases together we have that for any relay i it holds

$$\mathcal{V}_t[i] - \mathcal{V}'_{t-1}[i] \geq \frac{e}{2d} \cdot \min\{r, \tilde{r}\} - O(e/n)\,.$$

By Lemma 3.18 there are at least $|S|/2$ stations such that $\min\{r, \tilde{r}\} \geq d/2$. This implies that for each of these stations we have $\mathcal{V}_t[i] - \mathcal{V}'_{t-1}[i] \geq e \cdot d/4 - O(e/n)$. Therefore summing up over all $|S|/2$ such stations we obtain

$$|\mathcal{V}_t| - \left|\mathcal{V}'_{t-1}\right| \geq \Omega(e \cdot n)\,.$$

■

In contrast to the previous lemma, we now investigate how much the measure $\mathcal{V}$ may decrease when *Go-To-The-Middle* is applied. The result is that it is only by a constant factor.

Lemma 3.20. *Let the Go-To-The-Middle strategy be executed in time step t. Then it holds*

$$|\mathcal{V}_t| - \left|\mathcal{V}'_t\right| \leq \mathcal{R}$$

Proof. Let D_t denote the distance between the line L_t and the first station not included in S, i.e. $v_{|S|+1}$. The following link between $\mathcal{V}'$ and $\mathcal{V}$ is analogical to that established in Lemma 3.4 for the measure $\mathcal{X}$ and $\mathcal{Y}$

$$\mathcal{V}'_t[i] = \begin{cases} 0 & \text{if } i = 0 \\ 1/2 \cdot (\mathcal{V}_t[i-1] + \mathcal{V}_t[i+1]) & \forall i \in [1, \ldots, |S|-1] \\ 1/2 \cdot (D_t + \mathcal{V}_t[|S|-1]) & \text{if } i = |S|\,. \end{cases}$$

Therefore we have

$$\begin{aligned}
\left|\mathcal{V}'_t\right| &= \sum_{i=1}^{|S|} \mathcal{V}'_t[i] = \tfrac{1}{2}\left(D_t + \mathcal{V}_t[|S|-1]\right) + \\
&\quad \sum_{i=1}^{|S|-1} \tfrac{1}{2}\left(\mathcal{V}_t[i-1] + \mathcal{V}_t[i+1]\right) \\
&= \tfrac{1}{2}\left(D_t + \mathcal{V}_t[|S|-1]\right) + \sum_{i=1}^{|S|-2} \tfrac{1}{2}\cdot\mathcal{V}_t[i] + \sum_{i=2}^{|S|} \tfrac{1}{2}\cdot\mathcal{V}_t[i] \\
&= \tfrac{1}{2}\left(D_t + \mathcal{V}_t[1] + \mathcal{V}_t[|S|]\right) + \sum_{i=2}^{|S|-1} \mathcal{V}_t[i] \\
&= |\mathcal{V}_t| - \tfrac{1}{2}\cdot\left(V_t[1] + V_t[S] - D_t\right) \geq |\mathcal{V}_t| - \mathcal{R}\,,
\end{aligned}$$

since $\mathcal{V}_t[1] \leq \mathcal{R}$ and $\mathcal{V}_t[S] - D_t \leq \mathcal{R}$. ■

After introducing bounds on the change of the measure $\mathcal{V}$ we are ready to prove the upper bound on the average speed of the explorer.

Proof of Theorem 3.15. We construct the input sequence by letting the explorer move by a fixed vector e in clockwise direction around the base camp. We choose the start configuration so that all relays are on their optimal positions, so that $|\mathcal{V}_1| = 0$. Note that it holds

$$|\mathcal{V}_{t+1}| = |\mathcal{V}_t| + |\mathcal{V}_{t+1}| - \left|\mathcal{V}'_t\right| - \left(|\mathcal{V}_t| - \left|\mathcal{V}'_t\right|\right).$$

Therefore by Lemma 3.19 and 3.20 it holds

$$|\mathcal{V}_{t+1}| = |\mathcal{V}_1| + t\cdot\Omega(e\cdot n) - t\cdot\mathcal{R}\,.$$

As by Eq. (3.9) the value of $|\mathcal{V}_t|$ is upper bounded and therefore

$$t\cdot\Omega(e\cdot n) - t\cdot\mathcal{R} \leq n^2\cdot\mathcal{R}\,.$$

Thus, for $t \geq n^2\cdot\mathcal{R}$ it holds $e = \mathcal{O}(\mathcal{R}/n) = \mathcal{O}(1/d)$. ■

3.5. Imprecise Sensoric Data

Up to now we have assumed that the local coordinate system of the relays is precise. We now want to investigate how *Go-To-The-Middle* behaves with imprecise sensory input. For that purpose we assume an adversarial model, where an adversary biases the positions of neighbors in the local coordinate system of a relay. We stick to the static scenario with an immobile explorer, so that once again $n := n_t$.

For the purpose of analysis we will assume a global coordinate system in which the positions of the relays are expressed. During the execution of the *Go-To-The-Middle* strategy, we assume that the adversary constructs the local coordinate system of each relay, being allowed to change the positions of the neighbors v_{i-1} and v_{i+1} by at most ϵ from their true position in the global coordinate system.

We start with all relays aligned optimally on the line segment between the explorer and the base camp. We assume that both explorer and base camp are placed on the X-axis of the global coordinate system. Let $(x_{-1}; y_{-1}) = p_t(i-1)$ and $(x_{+1}; y_{+1}) = p_t(i+1)$. We let the adversary always change the position of the neighbors so that v_i perceives the position of v_{i-1} in time step at $(x_{-1}; y_{-1} + \epsilon)$ and the position of v_{i+1} at $(x_{+1}; y_{+1} + \epsilon)$. Therefore, station v_i moves in time step t to position

$$p_{t+1}(i) = \left(\frac{x_{-1} + x_{+1}}{2}; \frac{y_{-1} + y_{+1}}{2} + \epsilon\right).$$

We reuse the $\mathcal{Y}_t$ measure. We divide the execution of the *Go-To-The-Middle* strategy into conceptually two steps – we assume that relays first move to the positions they would have computed if they had precise information about the location of their neighbors. Afterward, all relays move by ϵ to reach the positions defined earlier. Therefore, we have $\mathcal{Y}''$ defined as earlier in Lemma 3.4 and account the movement by ϵ to $\mathcal{Y}_{t+1}[i] - \mathcal{Y}'_t[i] = \epsilon$. Applying these errors to all relays we obtain the following corollary.

Corollary 3.21. *The adversary can dictate errors in a way such that*

$$\left|\mathcal{Y}_{t+1}\right| - \left|\mathcal{Y}'_t\right| \geq n \cdot \epsilon \ .$$

As all relays are at the line connecting the explorer and base camp at the beginning and we are moving them on one side of this line we may assume that $\mathcal{Y}_t \geq 0$ for all t. This allows us to apply Lemma 3.5 and state the following corollary.

Corollary 3.22. *It holds*

$$\left|\mathcal{Y}_t\right| - \left|\mathcal{Y}'_t\right| \leq \max_{i,j=1,\ldots,n} \left|\mathcal{Y}_t[i] - \mathcal{Y}_t[j]\right| \ .$$

Using Corollary 3.22 and Corollary 3.21 we may conclude that for any t

$$\begin{aligned} \left|\mathcal{Y}_{t+1}\right| - \left|\mathcal{Y}_t\right| &= \left(\left|\mathcal{Y}_{t+1}\right| - \left|\mathcal{Y}'_t\right|\right) - \left(\left|\mathcal{Y}_t\right| - \left|\mathcal{Y}'_t\right|\right) \\ &\geq \epsilon \cdot n - \max_{i,j=1,\ldots,n} \left|\mathcal{Y}_t[i] - \mathcal{Y}_t[j]\right| \end{aligned}$$

Therefore the measure $\mathcal{Y}_t$ increases by at least $\epsilon/2 \cdot n$ as long as

$$\max_{i,j=1,\dots,n} \left|\mathcal{Y}_t[i] - \mathcal{Y}_t[j]\right| \leq \epsilon/2 \cdot n$$

Thus, in some time step k we have

$$\max_{i,j=1,\dots,n} \left|\mathcal{Y}_k[i] - \mathcal{Y}_k[j]\right| \geq \epsilon/2 \cdot n\,,$$

implying that the chain is surely disconnected. This shows, that *Go-To-The-Middle* can behave unstable if the sensors are subject to errors dictated by an adversary.

3.6. Strategies Similar to Go-To-The-Middle

Using the framework introduced for the analysis of the *Go-To-The-Middle* strategy we may also try to analyze similar strategies. Although we are not able to provide a general lower bound for a class of strategies similar to *Go-To-The-Middle*, we may give some intuitions why certain approaches will lead to strategies having performance similar to those of *Go-To-The-Middle*.

Go-Over-Middle

The *Go-Over-Middle* strategy also uses the point in the middle between a relay's neighbors as a reference. Though, the relay applying *Go-Over-Middle* does not stop at this middle point, but continues moving for a certain distance. Formally, let p' denote the point in the middle of the line segment $\langle p_t(i-1), p_t(i+1)\rangle$. The vector $\vec{u}$ has its initial point in $p_t(i)$ and its terminal point p'. In the *Go-Over-Middle* strategy with factor c, the relay v_i multiplies the vector $\vec{u}$ with c and moves by $c\vec{u}$. Therefore, it moves "over the middle". The factor $c \geq 1$ for a reasonable strategy. In the following theorem we show that the *Go-Over-Middle* strategy requires $\Omega(n^2)$ time to shorten the chain in the static scenario.

Theorem 3.23. *There exists a communication-valid and movement-valid chain of relays such that the Go-Over-Middle strategy with factor c must execute* $\Omega(\frac{1}{c}n^2)$ *steps in order to obtain* $|\mathcal{V}_t[i]| \leq 1$ *for all relays* $i = 1, \dots, n$.

We utilize the $\mathcal{V}_t$ measure, introduced in Section 3.4, to describe the distance of relays to the line L_t. Similarly as earlier, let $\mathcal{V}'_t$ describe the distances of relays to line L_t after *Go-Over-Middle* has been executed in time step t.

Lemma 3.24. *Let the Go-Over-Middle strategy with factor c by executed in time step t. Then it holds*

$$|\mathcal{V}_t| - \left|\mathcal{V}'_t\right| \leq c \cdot \mathcal{R} .$$

Proof. Assume that we have applied the *Go-To-The-Middle* strategy in time step t. Let $\mathcal{W}_t$ denote the distances of relays to L_t after *Go-To-The-Middle* has been applied. Then by Lemma 3.20 we have

$$|\mathcal{V}_t| - |\mathcal{W}_t| \leq \mathcal{R} .$$

As the relay executing *Go-Over-Middle* moves c times the distance as a relay executing *Go-To-The-Middle* we have

$$\mathcal{V}_t[i] - \mathcal{V}'_t[i] = c(\mathcal{V}_t[i] - \mathcal{W}_t[i]) .$$

Therefore we have

$$|\mathcal{V}_t| - \left|\mathcal{V}'_t\right| = \sum_{i=1}^{n} (\mathcal{V}_t[i] - \mathcal{V}'_t[i]) = c\left(\sum_{i=1}^{n} \mathcal{V}_t[i] - \mathcal{W}_t[i]\right) = c\,(|\mathcal{V}_t| - |\mathcal{W}_t|) \leq c \cdot \mathcal{R} .$$

■

By the former lemma the measure $\mathcal{V}_t$ decreases by at most $c \cdot \mathcal{R}$ per time step. Starting with a chain with $|\mathcal{V}_1| = \Omega(n^2)$ we need $O(\frac{1}{c}n^2)$ time steps to reduce the measure $\mathcal{V}$ to $O(n)$. Therefore, using the same construction as in Theorem 3.6 we may prove our Theorem 3.23.

In Chapter 4 we will show that the *Go-Over-Middle* strategy can perform better, if certain side conditions are fulfilled.

Biased Go-To-The-Middle

In the *Go-To-The-Middle* strategy the relays always move to the middle point of the line segment between their neighbors. We can modify this approach and define a strategy where the relay is moved to another point on this line segment. This may seem advantageous, as by biasing the relays to move toward the base camp in the chain we may hope to gather unnecessary relays at the base camp.

For our analysis we assume a fixed bias coefficient $c \in (0, 1)$. Formally, the target point for relay v_i in time step t lies on the line segment $\langle p_t(i-1), p_t(i+1)\rangle$ in distance $c\left|p_t(i-1) - p_t(i+1)\right|_2$ from $p_t(i+1)$.

Unfortunately, we can show that biasing does not help in achieving a better performance than $\Omega(n^2)$ time steps in the static scenario, as shown by the following theorem.

Theorem 3.25. *There exists a communication-valid and movement-valid chain of relays such that the Go-To-The-Middle strategy with bias coefficient c must execute* $\Omega(\frac{1}{c}n^2)$ *steps in order to obtain* $|\mathcal{V}_t[i]| \leq 1$ *for all relays* $i = 1, \ldots, n$.

We will once again use the $\mathcal{V}_t$ measure for measuring the progress of the strategy.

Lemma 3.26. *Let the Go-To-The-Middle strategy with bias coefficient c by executed in time step t. Then it holds*

$$|\mathcal{V}_t| - \left|\mathcal{V}'_t\right| \leq \mathcal{R}\,.$$

Proof. The bias in the *Go-To-The-Middle* strategy causes the $\mathcal{V}'_t$ vector to depend on $\mathcal{V}_t$ in a slightly different way than in the usual *Go-To-The-Middle* strategy.

$$\mathcal{V}'_t[i] = \begin{cases} 0 & \text{if } i = 0 \\ (1-c)\cdot\mathcal{V}_t[i-1] + c\cdot\mathcal{V}'_t[i+1] & \forall i \in [1,\ldots,n] \\ 0 & \text{if } i = n+1\,. \end{cases}$$

Therefore we have

$$\begin{aligned} \left|\mathcal{V}'_t\right| &= \sum_{i=1}^{n} \mathcal{V}'_t[i] \\ &\quad \sum_{i=1}^{n} (1-c)\cdot\mathcal{V}_t[i-1] + c\cdot\mathcal{V}_t[i+1] \\ &= \sum_{i=1}^{n-1} (1-c)\cdot\mathcal{V}_t[i] + \sum_{i=2}^{n} c\cdot\mathcal{V}_t[i] \\ &= (1-c)\cdot\mathcal{V}_t[1] + c\cdot\mathcal{V}_t[n] + \sum_{i=2}^{n-1} \mathcal{V}_t[i] \\ &= |\mathcal{V}_t| - (1-c)\cdot\mathcal{V}_t[1] - c\cdot\mathcal{V}_t[n] \geq |\mathcal{V}_t| - \mathcal{R}\,, \end{aligned}$$

since $\mathcal{V}_t[1] \leq \mathcal{R}$ and $\mathcal{V}_t[n] \leq \mathcal{R}$. ■

By the former lemma the measure $\mathcal{V}_t$ decreases by at most $c\cdot\mathcal{R}$ per time step. Similarly as for the *Go-Over-Middle* strategy we can show that we need $\Omega(n^2)$ time steps to reduce the measure $\mathcal{V}_t$ to $O(n)$. This proves Theorem 3.25.

CHAPTER 4

Hopper Strategies

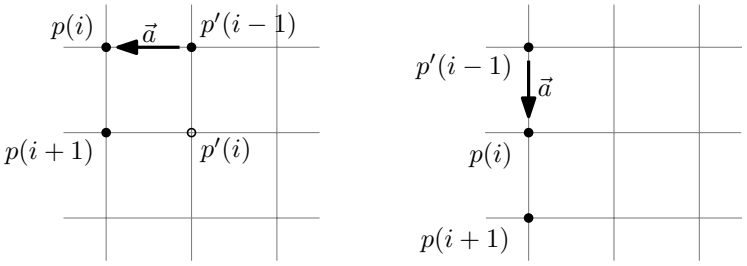
In this chapter we will present the *Hopper* strategy. For a better understanding of the concepts underlying the *Hopper* strategy, we will first discuss the *Manhattan-Hopper* strategy, which is very similar to *Hopper* but works with a slightly simplified model. In that model we assume all stations to move on a grid and not on a continuous plane. The relays used are stateless and work with precise local coordinate systems. Both *Manhattan-Hopper* and *Hopper* strategies are removing relays from the chain when it becomes shorter – as we will see, this allows for a very good performance of both strategies and gives a real advantage in comparison to the *Go-To-The-Middle* strategy. In concrete terms, we will show that in the static scenario the *Manhattan-Hopper* and *Hopper* strategies are able to achieve an optimal chain length in time linear in the number of relays.

The basic movement type executed by the *Manhattan-Hopper* and *Hopper* strategies is that of the *Go-Over-Middle* strategy, presented in Section 3.6, with factor 2. Though there are two significant differences between these strategies which we account to be responsible for its improved performance. That is the removal of unnecessary relays and the alignment of relays on grid points.

Both strategies in this chapter work in sequential runs, rather than completely in parallel as the *Go-To-The-Middle* strategy. In a run the actions are executed by relays in the chain one after another. We will see that this model of execution improves the performance of strategies and is very convenient for our analysis.

4.1. The Manhattan-Hopper Strategy

We begin our investigations with introducing the model for the *Manhattan-Hopper* strategy and the strategy itself.



(a) Stations not in line, v_i moves

(b) Stations in line, v_i does not move

Figure 4.1.: Hop-operation in *Manhattan-Hopper* strategy

In contrast to the *Go-To-The-Middle* strategy, *Manhattan-Hopper* is designed to work on a grid laid upon the plane. For the strategy to operate properly, all stations have to assume positions on the grid points. Therefore, we restrict the movement freedom of both the relays and the explorer, by forcing them to move between grid points. The mesh size for the grid is fixed to $\frac{1}{2}\mathcal{R}$.

In order to guarantee all stations positioned on the grid, we assume that in the first time step all positions of stations are on grid points and that the explorer may only move between grid points.

Relays are given precise local coordinate systems and precise odometry.

Strategy Description

The *Manhattan-Hopper* strategy is executed in *runs*. In each run, a relay v_i executes its action only after relay v_{i-1} has finished its own. We will see later that such runs can be pipelined, so that a new run can be started every three steps. For the sake of analysis it is sufficient and more comfortable to assume that a new run is executed only after the previous one has finished. We will stick to that assumption for the rest of the chapter.

We say that the chain is in *proper condition* if each two neighbored stations in the chain are positioned in neighbored points of the grid. For this discrete scenario it is natural to measure the distance of the chain in terms of its manhattan length. We say that the chain has optimal length if its length is equal to the manhattan distance between the explorer and base camp.

Let us first describe the *Manhattan-Hopper* strategy for the setting of the immobile explorer. We will extend it to accommodate explorer's movement in Section 4.1.2. As already mentioned, the *Manhattan-Hopper* strategy is executed in runs. For the sake of our description, let us fix a specific run *r*. We will use a different notation for the positions of runs than in the analysis of the *Go-To-The-Middle* strategy. Thus, let $p(i)$

denote the position of v_i at the beginning of this run, while $p'(i)$ is its position after it has executed the *Manhattan-Hopper* strategy.

In a run, the relay v_i moves only after v_{i-1} has finished its movement and prior to the movement of v_{i+1}. Therefore, the positions of v_i's neighbors are $p'(i-1)$ and $p(i+1)$ when v_i decides on its action. Typically, the relay v_i executes the *hop-operation*, defined as follows. Let $\vec{a}$ be a vector such that $p'(i-1) + \vec{a} = p(i)$ as shown in Fig. 4.1. Then the hop-operation executed by relay v_i moves it to position $p'(i)$, such that $p'(i) = p(i+1) - \vec{a}$. This implies that, if v_{i-1}, v_i, v_{i+1} lie in-line, then station v_i does not move. This situation is depicted in Fig. 4.1(b). The provided definition of the hop-operation is equivalent to mirroring v_i's position w.r.t. the line through $p'(i-1)$ and $p(i+1)$.

A special case occurs when relay v_i moves to a point on the grid occupied by v_{i+2}. Then a *remove*-operation is executed, i.e., v_{i+1} and v_{i+2} are removed from the chain and the run stops. After this removal, we index the chain once again.

Thus, a run of the *Manhattan-Hopper* strategy either removes two relays from the chain, or is executed by all relays. Removing relays in this way is the only way how the *Manhattan-Hopper* strategy shortens the length of the chain. In particular, if v_i moves to a point already occupied by a relay v_j such that $j \neq i+2$, no relays are removed and the execution of the run proceeds as earlier. This may sound counterproductive as the chain part forming a loop between v_i and v_j might be removed. In our analysis, we assume that such loops are not removed, because this would require information exchange among the relays, those in the loop have to get to know that they are obsolete. It is easy to see that the same results as we show also hold in the stronger model, where long loops may be removed.

A reasonable performance of the *Manhattan-Hopper* strategy can only be attained if the runs are not executed one after another but pipelined in a fast manner. This is possible without problems due to the sequential nature of the runs of the *Manhattan-Hopper* strategy. Run r starting execution in time step t may be followed by run $r+1$ starting execution in time step $t+3$. Thus, the execution of m runs starting with a chain with n relays takes at most $n + 3m$ time steps.

4.1.1. Performance in the Static Scenario

We first investigate the performance of the *Manhattan-Hopper* strategy in the static case and eventually show the following theorem.

Theorem 4.1. *Starting with a chain in proper condition with n relays, the Manhattan-Hopper strategy ensures that*

(a) *the chain remains in proper condition,*

(b) *the relays move at most distance $\frac{1}{2}\sqrt{2}\cdot\mathcal{R}$ per run, i.e., every third step, and*

(c) *after n runs, i.e., 4n steps, the chain has optimal length.*

Property (b) follows directly from the definition of the strategy. Also property (a) is easy to see: If a run does not execute a removal, then the manhattan distances between neighbors do not change. Whenever stations v_{i+1} and v_{i+2} are removed from the chain, relay v_i is on the position of v_{i+2} and therefore has manhattan distance $\frac{1}{2}\mathcal{R}$ to v_{i+3}.

We will now show that n runs suffice to reduce the chain to optimal length. As we can start a new run every 3 steps, this implies (c).

Let d denote the (manhattan) distance between the explorer and the base camp. Fix some run and let $p(i)$ and $p'(i)$ denote the positions of the station v_i before and after this run, respectively.

We can describe the positions of the relays with respect to the explorer using the spatial unit vectors $\vec{e_0} = \left[\frac{1}{2}\mathcal{R}, 0\right]$, $\vec{e_1} = \left[-\frac{1}{2}\mathcal{R}, 0\right]$, $\vec{e_2} = \left[0, \frac{1}{2}\mathcal{R}\right]$, $\vec{e_3} = \left[0, -\frac{1}{2}\mathcal{R}\right]$. $\vec{e_0}$ and $\vec{e_1}$ are both horizontal vectors, with opposite directions. Analogously, $\vec{e_2}$ and $\vec{e_3}$ are vertical. We will call such pairs of vectors *oppositional*. For our run, we can define a sequence of spatial unit vectors $C = (\vec{u_0}, \vec{u_1}, \ldots, \vec{u_k})$ such that $p(i) = p(0) + \vec{u_0} + \cdots + \vec{u_{i-1}}$. This sequence will be called *configuration* and it describes the positions of all relays in the chain in relation to the explorer by using the unit vectors.

We want to describe the behavior of *Manhattan-Hopper* in terms of operations on configurations. The following two lemmas describe how one run of the *Manhattan-Hopper* strategy can modify the configuration.

Lemma 4.2. *Let $C = (\vec{u_0}, \vec{u_1}, \ldots, \vec{u_k})$ be the configuration at the beginning of the run. Assume the run finishes without removing any relays from the chain. Then the configuration at the beginning of the next run is $C' = (\vec{u_1}, \ldots, \vec{u_k}, \vec{u_0})$. Furthermore, for each $j = 0, \ldots, k$, the vectors $\vec{u_0}$ and $\vec{u_j}$ are not oppositional.*

Proof. We will show that the invariant $p'(i) = p(i+1) - \vec{u_0}$ holds for all $i = 0, \ldots, k$ by induction. The induction basis is clearly given, as we have by definition of vector $\vec{u_0}$ that $p'(0) + \vec{u_0} = p(0) + \vec{u_0} = p(1)$.

Assume now that the invariant holds for all $i \leq j$. Then we have $p'(j) = p(j+1) - \vec{u_0}$. Consider the hop-operation performed by relay v_{j+1}. As the position $p'(j+1)$ depends on the vector between $p'(j)$ and $p(j+1)$ and the position of v_{j+2} we have from the definition of the hop-operation $p'(j+1) = p(j+2) - \vec{u_0}$. This proves the invariant.

The positions of all relays v_i for $i = 1, \ldots, k$ at the beginning of the run are, by the definition of the configurations, given by

$$p(i) = p(0) + \vec{u}_0 + \cdots + \vec{u}_{i-1} .$$

After the run we have by our invariant for all relays

$$p'(i) = p(i+1) - \vec{u}_0 = \vec{u}_1 + \cdots + \vec{u}_i .$$

Obviously the base camp does not move and therefore we have $p'(k+1) = p(k+1)$. Thus,

$$p'(k+1) = p(k) + \vec{u}_0 = \vec{u}_1 + \cdots + \vec{u}_k + \vec{u}_0 .$$

The final configuration is thus given by $C_{r+1} = (\vec{u}_1, \ldots, \vec{u}_k, \vec{u}_0)$.

We will proceed with proving the second statement of the lemma. Assume for the sake of contradiction that there exists a vector $\vec{u}_j$ which is oppositional to $\vec{u}_0$, i.e. $\vec{u}_j = -\vec{u}_0$. By our earlier observation we have $p'(j-1) = p(j) - \vec{u}_0$ after the run strategy has been executed by v_j. Therefore $p'(j-1) = p(j) - \vec{u}_0 = p(j) + \vec{u}_j = p(j+1)$, as by its definition vector $\vec{u}_j$ connects v_j to v_{j+1}. In that case, the run would remove relays v_j and v_{j+1} from the chain, contradicting the assumptions of our lemma. ■

Lemma 4.3. *Let $C = (\vec{u}_0, \vec{u}_1, \ldots, \vec{u}_k)$ be the configuration at the beginning of the run. The run finishes with removing v_{i+1} and v_{i+2} from the chain if and only if the vector $\vec{u}_{i+1}$ is the first vector in C oppositional to $\vec{u}_0$. The configuration at the beginning of the next run is then given by $C' = (\vec{u}_1, \ldots, \vec{u}_i, \vec{u}_{i+2}, \ldots, \vec{u}_k)$.*

Proof. We will first show that, if v_{i+1} and v_{i+2} are removed from the chain, then $\vec{u}_{i+1}$ is oppositional to $\vec{u}_0$. Since all relays prior to v_i execute the hop-operation without removing any relays from the chain, the execution of the run for these relays is the same as the execution described in Lemma 4.2. Therefore, for all $j \leq i$ it holds that

$$p'(j) = \vec{u}_1 + \cdots + \vec{u}_j .$$

As v_{i+1} and v_{i+2} have been removed, it must hold $p'(i) = p(i+2)$. Therefore

$$p'(i) = \vec{u}_1 + \cdots + \vec{u}_i = \vec{u}_0 + \vec{u}_1 + \cdots + \vec{u}_{i+1} = p(i+2) ,$$

so that $\vec{u}_0 = -\vec{u}_{i+1}$.

For the proof in the other direction assume that $\vec{u}_{i+1}$ is the first vector in C oppositional to $\vec{u}_0$. By the previous argument, no relay pair v_{j+1}, v_{j+2} with $j < i$ can be removed, as then $\vec{u}_{j+1}$ would be oppositional to $\vec{u}_0$. Therefore all relays v_j for $j \leq i$ have executed the hop-operation, so that

$$p'(i) = \vec{u}_1 + \cdots + \vec{u}_i .$$

As $\vec{u}_0 = -\vec{u}_{i+1}$ it holds

$$p'(i) = \vec{u}_1 + \cdots + \vec{u}_i = \vec{u}_0 + \vec{u}_1 + \cdots + \vec{u}_{i+1} = p(i+2)\,,$$

and v_{i+1} and v_{i+2} are removed by the *Manhattan-Hopper* strategy.

Assume now that v_{i+1} and v_{i+2} are removed from the chain in the current run. From our previous reasoning we have for any $j \leq i$

$$p'(j) = \vec{u}_1 + \cdots + \vec{u}_j$$

and for all $j \geq i+3$

$$p'(j) = \vec{u}_1 + \cdots + \vec{u}_i + \vec{u}_{i+2} + \cdots + \vec{u}_{j-1}\,.$$

The configuration after the completed run is therefore given by

$$C_{r+1} = (\vec{u}_1, \ldots, \vec{u}_i, \vec{u}_{i+2}, \ldots, \vec{u}_k)\,.$$

■

Let $C_1 = (\vec{a}_0, \ldots, \vec{a}_k)$ be the configuration describing the chain at the beginning of the first run. Each of the vectors $\vec{a}_0, \ldots, \vec{a}_k$ is of the type of one of the unit vectors, nevertheless we can keep track of each of them by assigning them unique identification numbers. The two former lemmas show that each run of *Manhattan-Hopper* either removes vectors from this configuration or the configuration is shifted cyclically. Particularly, no vector is added to the configuration by the *Manhattan-Hopper* strategy. Therefore, we can describe each configuration occurring in the future as some arrangement of a subset of the vectors $\vec{a}_0, \ldots, \vec{a}_k$.

Thus, we can follow the trace of each $\vec{a}_j$, whose initial position in the start configuration is j. Assume that, in run r, it has position l. Then it is either removed (by Lemma 4.3), or its position is reduced by 1 (in case of the shift in Lemma 4.2 or in case of Lemma 4.3, if the removed relay is v_i with $i > l$) or its position is reduced by 2 (in case of Lemma 4.2, if the removed relay is v_i with $i < l$).

Now assume that, after n runs, there still exists an oppositional pair of vectors $\vec{u}_i, \vec{u}_j$, with $i < j$. Then, by the above, at most p runs earlier, $\vec{u}_i$ was at position 0. As also $\vec{u}_j$ is in this configuration, $\vec{u}_i$ will be removed in the next run by Lemma 4.3, contradicting the assumption that $\vec{u}_i$ is part of the configuration after n runs.

Thus, there are no oppositional vectors in the configuration after n runs. This implies that the length of the path after n runs is minimal, namely the manhattan distance between the explorer and the base camp. This proves part (c) of the theorem.

4.1.2. Performance in the Dynamic Scenario

In order to adapt *Manhattan-Hopper* to work in a dynamic scenario, we have to deal with the movement of the explorer. For that purpose, we allow the explorer to move only to a grid point adjacent to its current position at once. The *Dynamic-Manhattan-Hopper* strategy groups the movement of the explorer and the execution of two runs into one round. The first run is a *follow*-run in which relays execute the *follow*-operation. This allows them to catch up with the movement of the explorer. Afterward a run of the *Manhattan-Hopper* strategy, a so called *hopper*-run is executed. At this point of time we assume that the two runs are executed one after another and the round only finishes when the runs are ready. We will elaborate more on the pipelining of the runs later.

The follow-operation is applied by relay v_i whenever v_{i-1} increases its distance to v_i to $\mathcal{R}$. Then relay v_i moves to the old position of the station v_{i-1}, i.e. $p'(i) = p(i-1)$. Thereby, the chain shifts itself in direction of the explorer by one grid point, so that finally the last relay is in manhattan distance $\mathcal{R}$ to the base camp. At this point of time, the base camp inserts a new relay at the old position $p(k)$ of the relay v_k. During the execution of the follow-operation the maximum distance between neighbored stations is $\mathcal{R}$ and therefore the chain remains connected. If the chain was in proper condition before the *follow*-run, it is so afterward, too.

While the follow-run ensures that the manhattan distance between two relays is exactly $\frac{1}{2}\mathcal{R}$, the hopper-run is meant to decrease the length of the chain if necessary. We first show that these two runs between every movement of the explorer are sufficient to maintain an optimal chain length.

Lemma 4.4. *Let the chain have optimal length prior to the explorer's movement. Then after the explorer's movement the follow-run and the hopper-run bring the chain to an optimal length.*

Proof. Let $C = (\vec{u}_0, \ldots, \vec{u}_k)$ be the configuration of the chain prior to explorer's movement. Let the explorer move by $\vec{u}$, i.e. $p'(0) = p(0) + \vec{u}$. Then, after the explorer's movement and the first run of the *Manhattan-Hopper* strategy we have $C' = (\vec{u}, \vec{u}_0, \ldots, \vec{u}_k)$. The manhattan length of the chain defined by C' increases with respect to C by $\frac{1}{2}\mathcal{R}$.

Recall that we are assuming that the length of the chain defined by C is optimal before the explorer's movement. When moving the explorer by $\vec{u}$ increases its manhattan distance to the base camp, then the length of the chain after applying the follow-run is already optimal and the hopper-run is actually unnecessary.

In case the explorer's movement decreases its manhattan distance to the base camp, this decrease is equal to $\frac{1}{2}\mathcal{R}$. We have to show that the hopper-run reduces the length of the chain by the same amount. Observe that, since the length of the chain described

by C' is not optimal, there must exist a pair of oppositional vectors in C'. On the other hand, there was no such pair in C. This implies, that there exists a vector $\vec{u}_j \in C$ which is oppositional to $\vec{u}$. By Lemma 4.3 the existence of a vector oppositional to the first vector of the configuration leads to a hopper-run which reduces the manhattan length of the chain by $\mathcal{R}$. ■

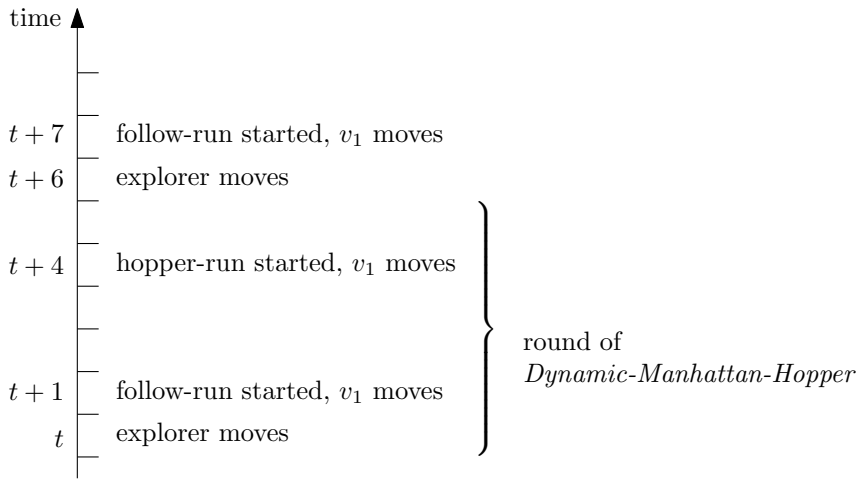


Figure 4.2.: One round of the execution of *Dynamic-Manhattan-Hopper*

Pipelined execution

The runs of the *Dynamic-Manhattan-Hopper* strategy are executed in the discrete, synchronized time model. In order to pipeline them, we have to make sure that consecutive runs do not interfere. This is guaranteed if the follow-run starts directly after the movement of the explorer, but two idle steps are inserted before the hopper-run and between the hopper run and the next movement, see Figure 4.2.

Thus the explorer may move every 6 time steps. Such a sequence of 6 time steps starting with a movement of the explorer is called a *round*.

Let d_r denote the distance between the explorer and the base camp at the beginning of round r. We are now ready to show the following theorem.

Theorem 4.5. *Assume we start with an optimal chain. Then, the chain maintained by the Dynamic-Manhattan-Hopper strategy has the following properties before each round r.*

(a) *The chain remains connected,*

(b) *the explorer may move a distance of $\frac{1}{2}\mathcal{R}$ every round, i.e. every 6th time step,*

(c) *relays move at most distance* $\frac{1}{2}\left(1+\sqrt{2}\right)\mathcal{R}$ *per round, and*

(d) *the number of relays used in the chain is at most* $2\left(\frac{4}{3\mathcal{R}}d_r+1\right)$.

Property (a) follows easily from the fact that the follow-run decreases the distances between neighbored relays to $\frac{1}{2}\mathcal{R}$. During a round a relay executes the follow-run, moving for at most $\frac{1}{2}\mathcal{R}$ and a hopper-run, moving for at most $\frac{1}{\sqrt{2}}\mathcal{R}$. Property (c) follows.

In case the runs are pipelined the decrease of the chain length corresponding to the movement of the explorer is delayed until the run started after the movement is finished. In the worst case this can take as many time steps as many relays have been in the chain at the moment the explorer moved. This can mean that the distance of the explorer to the base camp decreases during some timespan, while the chain is still executing the various runs which will eventually decrease its length. Therefore, we have to investigate how large the difference between the chain length and the explorer's distance may become.

Assume the chain has optimal length at the beginning of time step 1. The runs are pipelined with an appropriate time separation and thus there is no difference for run $r+1$ whether it is started 3 time steps after the start of run r or after a full completion of run r. Therefore and by Lemma 4.4 a hopper-run started in round r works on a chain of length at most d_r.

Let us fix round r. The number of relays in the chain at the beginning of round r is bounded from above by $2d_r/\mathcal{R}$ plus the number of unfinished hopper-runs times 2. Therefore, we will look at how many hopper-runs are still unfinished at the beginning of round r. Take a round $z < r - \frac{2d_r/\mathcal{R}-2}{6}$. The hopper-run started in round z will last for at most $2d_z/\mathcal{R}+2$ time steps and thereby will will finish not later than at round $z + \frac{2d_z/\mathcal{R}+2}{6}$. Obviously $d_r \le d_z + \frac{1}{2}\mathcal{R}(r-z)$ as the explorer may only move by $\frac{1}{2}\mathcal{R}$ per round. We can bound

$$z + \frac{2d_z/\mathcal{R}+2}{6} < r - \frac{2d_r/\mathcal{R}-2}{6} + \frac{2d_z/\mathcal{R}+2}{6} \le r - \frac{1}{6}(r-z) \le r\,.$$

Therefore a hopper-run started in time step z is finished before round r. There are at most $\frac{2d_r/\mathcal{R}-2}{6}$ runs still unfinished in round r and thereby the number of relays in the chain is at most $d_r/\mathcal{R} + \frac{2}{3}(d_r/\mathcal{R}+1)$.

4.2. The Hopper Strategy

The *Hopper* strategy is an extension of the *Manhattan-Hopper* strategy, designed in such a way that it does not require the stations to be positioned on a discrete grid. We still require the relays to have precise local coordinate systems, nevertheless we do not use

any fixed grid structure in order to organize the relays. Furthermore, we allow the explorer to move freely, without being restricted to grid points.

As for the *Manhattan-Hopper* strategy, we will describe the chain at the beginning of a run by the configuration $C = (\vec{u}_0, \ldots, \vec{u}_k)$. Note that, since stations are not necessarily aligned to grid points, the vectors $\vec{u}_i$ are no longer restricted to unit vectors as in the *Manhattan-Hopper* strategy. For two vectors $\vec{u}_i$ and $\vec{u}_j$ we define the angle $\angle(\vec{u}_i, \vec{u}_j)$ as the smaller of the two angles created by anchoring $\vec{u}_j$ at the terminal point of $\vec{u}_i$, as shown on Fig. 4.3. Note that the $\angle$ function is symmetric, i.e. $\angle(\vec{u}_i, \vec{u}_j) = \angle(\vec{u}_j, \vec{u}_i)$ and thereby $\angle(\vec{u}_i, \vec{u}_j) \leq \pi/2$.

We will say that a vector $\vec{u}$ has a conflict to a vector $\vec{a}$ if $\angle(\vec{u}, \vec{a}) \leq \pi/2$. A vector which has angle larger than $\pi/2$ to all vectors from a configuration is called non-conflicting.

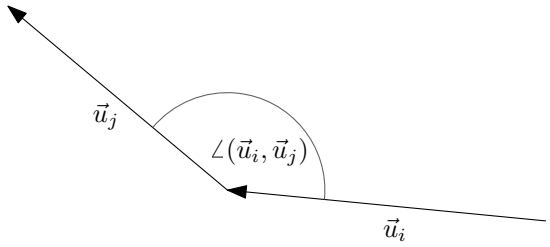


Figure 4.3.: Angle between vectors $\vec{u}_i$ and $\vec{u}_j$

The *Hopper* strategy is executed, exactly as *Manhattan-Hopper*, in sequential runs. Once again, let us fix a run. Then $p(i)$ and $p'(i)$ will denote the positions of the station v_i before and after this run, respectively.

The behavior of relay v_i in the run depends on whether the distance between $p'(i-1)$ and $p(i+1)$ is larger than $\mathcal{R}$ or not. If the distance is smaller or equal to $\mathcal{R}$ then obviously v_i is no longer necessary in the chain. In this case the relay removes itself, hereby executing the *remove-operation*. Executing the remove-operation finishes the run of the *Hopper* strategy.

In case the distance between $p'(i-1)$ and $p(i+1)$ is larger than 1, the action of v_i depends on the angle between the line segments $\langle p'(i-1), p(i)\rangle$ and $\langle p(i), p(i+1)\rangle$. The hop-operation is invoked if the angle is larger than $\pi/2$, otherwise the shorten-operation is used.

Hop-operation. The hop-operation is exactly the same as used previously in the *Manhattan-Hopper* strategy. Let $\vec{a}$ be a spatial vector such that $p(i-1) + \vec{a} = p(i)$. Then the hop-operation executed by relay v_i moves it to position $p'(i) = p(i+1) - \vec{a}$. This is depicted in Fig. 4.4. After the hop-operation has been executed, the sequential run proceeds at the next relay.

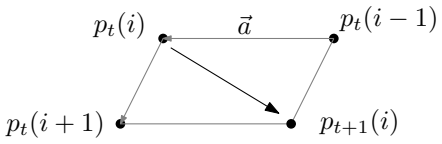


Figure 4.4.: Relay v_i executing the hop-operation in time step t

Shorten-operation. In the shorten-operation the relay v_i moves to the middle of the line segment between $p'(i-1)$ and $p(i+1)$, just as it did in the *Go-To-The-Middle* strategy. Executing the shorten-operation finishes the run of the *Hopper* strategy.

Again, we can pipeline runs, so that a new run starts in every third step.

4.2.1. Performance in the Static Scenario

We again denote by n the number of relays at the beginning of the first run and by d the distance between the explorer and the base camp.

The analysis of the performance of the *Hopper* strategy will proceed in a way similar to the analysis of the *Manhattan-Hopper* strategy. Unfortunately, the *Hopper* strategy not only removes or moves vectors in the configuration but also adds new vectors. In the course of the analysis we will show how these new vectors are formed. Eventually, we will show that after a linear number of steps, all pairs of vectors in the configuration form an angle large than $\pi/2$. Our last lemma will show that a chain described by such a configuration is short. This yields the following theorem.

Theorem 4.6. *Starting with a chain with n relays, the Hopper strategy ensures that*

(a) *the chain remains connected,*

(b) *relays move at most distance* $\sqrt{2} \cdot \mathcal{R}$ *per run,*

(c) *after* $8n + 1$ *runs, i.e.,* $25n + 1$ *steps, the chain uses at most* $2\sqrt{2}d/\mathcal{R} + 1$ *relays.*

The parts (a) and (b) are immediate from the strategy. The rest of this chapter is devoted to proving the bound on the number of runs stated in part (c).

Changes in configurations. We start with showing how a run can influence the configurations.

Lemma 4.7. *Let $C = (\vec{u}_0, \vec{u}_1, \ldots, \vec{u}_k)$ be the configuration at the beginning of a run. Assume the run finishes without executing the remove-operation or the shorten-operation. Then the configuration after the run has completed is $C' = (\vec{u}_1, \ldots, \vec{u}_k, \vec{u}_0)$. Furthermore, for each $i = 0, \ldots, k$, $\angle(\vec{u}_0, \vec{u}_i) > \pi/2$ holds.*

Proof. In the case of a run finishing without removing any relays from the chain and without executing the shorten-operation, all relays in the chain have been executing the hop-operation only.

The hop-operation is executed by relay v_i only if the angle between $\langle p'(i-1), p(i)\rangle$ and $\langle p(i), p(i+1)\rangle$ is larger than $\pi/2$. As $p'(i-1) = p(i) - \vec{u}_0$ when all relays prior to i have executed the hop-operation, the angle considered is equal to $\angle(\vec{u}_0, \vec{u}_i)$. This implies $\angle(\vec{u}_0, \vec{u}_i) > \pi/2$ for all $i = 0, \ldots, k$. ■

Lemma 4.8. *Let $C = (\vec{u}_0, \vec{u}_1, \ldots, \vec{u}_k)$ be the configuration at the beginning of the run. Assume the run finishes with v_i executing the remove-operation or the shorten-operation. Let $C' = (\vec{a}_0, \vec{a}_1, \ldots, \vec{a}_m)$ be the configuration of the chain at the beginning of the next run. Then each vector $\vec{a}_i$ is a positive linear combination of the vectors $\vec{u}_0, \ldots, \vec{u}_k$. More precisely, $C' = (\vec{u}_1, \ldots, \vec{u}_{i-1}, \frac{1}{2}(\vec{u}_0 + \vec{u}_i), \frac{1}{2}(\vec{u}_0 + \vec{u}_i), \vec{u}_{i+1}, \ldots, \vec{u}_k)$ in case of a shorten-operation, and $C' = (\vec{u}_1, \ldots, \vec{u}_{i-1}, \vec{u}_0 + \vec{u}_i, \vec{u}_{i+1}, \ldots, \vec{u}_k)$ in case of a remove operation.*

Proof. Assume that relay v_i executes the remove- or the shorten-operation. Then all relays with indices smaller than i execute the usual hop-operation in the run. We can adapt the invariant introduced for the analysis of the *Manhattan-Hopper* strategy and state that $p'(j) = p(j+1) - \vec{u}_0$ holds for all $j < i$.

Let us first assume that the run has ended with a shorten-operation. The configuration of the chain prior to executing the shorten-operation by v_i is $(\vec{u}_1, \ldots, \vec{u}_{i-1}, \vec{u}_0, \vec{u}_i, \ldots, \vec{u}_k)$. The shorten-operation moves relay v_i into the middle of the line segment between v_{i-1} and v_{i+1}. As v_{i-1} and v_{i+1} are connected through v_i, it holds $p'(i-1) + \vec{u}_0 + \vec{u}_i = p(i+1)$. Therefore the shorten-operation replaces the vectors $\vec{u}_0, \vec{u}_i$ in the configuration by two vectors $\frac{1}{2}(\vec{u}_0 + \vec{u}_i)$ and $\frac{1}{2}(\vec{u}_0 + \vec{u}_i)$. Thus

$$C' = (\vec{u}_1, \ldots, \vec{u}_{i-1}, \tfrac{1}{2}(\vec{u}_0 + \vec{u}_i), \tfrac{1}{2}(\vec{u}_0 + \vec{u}_i), \vec{u}_{i+1}, \ldots, \vec{u}_k)\,.$$

In the case the run ends with a relay being removed, the same argument yields

$$C' = (\vec{u}_1, \ldots, \vec{u}_{i-1}, \vec{u}_0 + \vec{u}_i, \vec{u}_{i+1}, \ldots, \vec{u}_k)\,.$$

■

Shorten-operations. If the run ends with removing a relay we are certainly improving the chain by reducing its length. We want to show that executing the shorten-operation also reduces the chain length by a significant amount, if the involved spatial vectors have a minimum length.

Let the length of a chain C be defined by

$$|C| = \sum_{i=0,\dots,k} |\vec{u}_i|.$$

Lemma 4.9. *Let $C = (\vec{u}_0, \vec{u}_1, \dots, \vec{u}_k)$ be the configuration at the beginning of the run. Assume the run finishes with v_i executing the shorten-operation, yielding configuration C'. If $|\vec{u}_0| \geq \frac{1}{2}\mathcal{R}$ and $|\vec{u}_i| \geq \frac{1}{2}\mathcal{R}$ then $|C'| \leq |C| - \frac{1}{3}\mathcal{R}$.*

Proof. The configuration of the chain prior to executing the shorten-operation by v_i is given by $(\vec{u}_1, \dots, \vec{u}_{i-1}, \vec{u}_0, \vec{u}_i, \dots, \vec{u}_k)$. For the sake of concise notation denote

$$a := \left|\langle p(i), p'(i-1)\rangle\right| = |\vec{u}_0|, \quad b := \left|\langle p(i), p(i+1)\rangle\right| = |\vec{u}_i|, \quad c := \left|\langle p'(i-1), p(i+1)\rangle\right| = |\vec{u}_0 + \vec{u}_i|\,.$$

As v_i moves to the middle of the line segment $\langle p'(i-1), p(i+1)\rangle$ the length of the chain decreases with the movement of v_i by $a + b - c$. Denote by γ the angle between $\langle p(i), p'(i-1)\rangle$ and $\langle p(i), p(i+1)\rangle$.

Note that by the Law of Cosines $c = \sqrt{a^2 + b^2 - 2ab\cos\gamma}$. Therefore, for fixed a, b, the value of c is maximized for $\gamma = \pi/2$. So, $a + b - c \geq a + b - \sqrt{a^2 + b^2}$. Using the border conditions $\frac{1}{2}\mathcal{R} \leq a, b \leq \mathcal{R}$ and $\mathcal{R} \leq c \leq \sqrt{a^2 + b^2}$ the function $a + b - \sqrt{a^2 + b^2}$ is minimized for $a = \frac{1}{2}\mathcal{R}, b = \frac{1}{2}\sqrt{3}\mathcal{R}$, with $a + b - \sqrt{a^2 + b^2} \geq \frac{1}{2}\mathcal{R}\left(\sqrt{3} - 1\right) \geq \frac{1}{3}\mathcal{R}$. ■

Unfortunately we cannot be sure that all vectors in the configuration have length at least $\frac{1}{2}\mathcal{R}$. Therefore there are some executions of the shorten-operation which shorten the path by an amount less than $\frac{1}{3}\mathcal{R}$.

Lemma 4.10. *There are at most $2n + 1$ executions of the shorten-operations such that one of the participating vectors has length smaller than $\frac{1}{2}\mathcal{R}$.*

Proof. There are at most $n + 1$ vectors with length smaller than $\frac{1}{2}\mathcal{R}$ at the beginning. The only possibility that a vector of length smaller than $\frac{1}{2}\mathcal{R}$ are created during the hopper-strategy is by the remove-operation. As such an operation can be executed at most n times, the lemma follows. ■

Non-conflicting suffix We now introduce a technical lemma, which is helpful in establishing the fact that a vector which becomes non-conflicting remains so for the whole future.

Lemma 4.11. *Let $\vec{u}$ be a spatial vector and $V = \vec{u}_1, \ldots, \vec{u}_k$ a set of spatial vectors. If $\angle(\vec{u}, \vec{u}_i) > \pi/2$ for all $i = 1, \ldots, k$, then, for any positive linear combination $\vec{s}$ of vectors from V, $\angle(\vec{u}, \vec{s}) > \pi/2$ holds.*

Proof. Without loss of generality let $\vec{u} = (\mathcal{R}, 0)$. Then for any vector $\vec{u}_i$, the condition $\angle(\vec{u}, \vec{u}_i) > \pi/2$ implies $\vec{u}_i = (a_i, b_i)$ with $a_i > 0$. Thus, each positive linear combination $\vec{u}'$ of the vectors from V is of the form $\vec{u}' = (a', b')$ with $a' > 0$. This implies that $\angle(\vec{u}, \vec{u}') > \pi/2$. ■

A non-conflicting suffix of a configuration C is a suffix of C consisting of non-conflicting vectors. Let S be the longest non-conflicting suffix of C. Then by Lemma 4.7, a run without remove- and shorten-operation makes S longer, because $\vec{u}_0$ is added to it.

Lemma 4.12. *The remove-operation and the shorten-operation do not decrease the size of the longest non-conflicting suffix.*

Proof. A shorten- or remove-operation can only shorten the non-conflicting suffix S, if it replaces $\vec{u}_0$ and some $\vec{u}_i$ contained in S by $\vec{u}_0 + \vec{u}_i$, or two copies of $\frac{1}{2}(\vec{u}_0 + \vec{u}_i)$. We will show that, in this situation, $\vec{u}_0 + \vec{u}_i$ is non-conflicting in the new configuration C'.

As $\vec{u}_i$ is included in S, it is non-conflicting in C. Furthermore, earlier in this run, a shorten-operation would have been executed if $\vec{u}_0$ had a conflict with a vector $\vec{u}_j$ with $j < i$. Therefore, also $\vec{u}_0$ is non-conflicting in C. This implies that $\vec{u}_0 + \vec{u}_i$ is by Lemma 4.11 non-conflicting and thus a proper replacement for $\vec{u}_i$ in S. Thus the size of the longest non-conflicting suffix does not decrease. ■

Taking together the results established so far we can describe the following possible effects of a run

1. it removes a relay,
2. it reduces the path length by at least $\frac{1}{3}\mathcal{R}$,
3. it reduces the path length by less than $\frac{1}{3}\mathcal{R}$,
4. it makes the non-conflicting suffix longer.

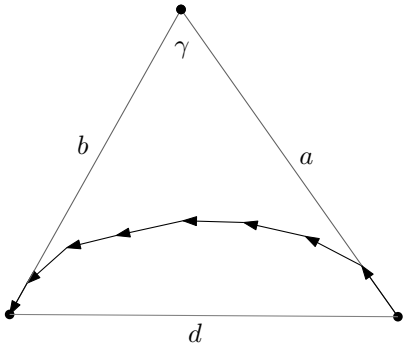


Figure 4.5.: The relay v_i is inside of the square with edge length d

Case 1 can only happen at most n times, Case 2 at most $3n$ times. By Lemma 4.10, Case 3 can occur only $2n + 1$ times. Case 4 occurs at most n times, because the length of the non-conflicting suffix can not exceed n, and it will never be shortened by Lemma 12.

Thus, after at most $6n + 1$ runs in total, the configuration is non-conflicting.

Chain length. What is still to be shown is that a non-conflicting configuration has small length. The following lemma proves this fact.

Lemma 4.13. *Let $C = (\vec{u}_0, \ldots, \vec{u}_k)$ be a non-conflicting configuration. Then*

$$|\vec{u}_0| + \cdots + |\vec{u}_k| \leq \sqrt{2}d\,.$$

Proof. Sort all the vectors from C ascending w.r.t. their angle to the line through the explorer and base camp, construct a configuration $C' = (\vec{a}_0, \ldots, \vec{a}_k)$ out of this order, and rearrange the relays so that they positions reflect the configuration C'. Obviously $|C| = |C'|$ although the shapes of the corresponding chains may differ significantly.

Due to the sorting of the vectors, the polygon defined by the configuration C' and the line segment between the explorer and the base camp is convex.

Take now $\vec{a}_0$ and $\vec{a}_k$ and enlarge both of them so that they form a triangle together with the line segment between the explorer and the base camp, as shown on Fig. 4.5. The length of C' is not larger than $a + b$ as C' is convex.

As $\angle(\vec{a}_0, \vec{a}_k) > \pi/2$, the angle γ is at least $\pi/2$. Note that for a fixed d, the sum $a + b$, expressed as a function of α is decreasing. Therefore, the sum $a + b$ is maximized for $\gamma = \pi/2$. Maximizing $a + b$ with the condition $\sqrt{a^2 + b^2} = d$ we obtain $a + b \leq \sqrt{2}d$. ■

Number of relays. The former lemma shows only that the length of the chain after $8n+1$ runs is bounded by $\sqrt{2}d$. We nevertheless have no bound on the number of relays used in this chain.

Lemma 4.14. *Let all k vectors be non-conflicting at the beginning of run r. Then after* $2k$ *runs at most one vector has length smaller than* $\frac{1}{2}\mathcal{R}$.

Proof. Since all vectors are non-conflicting, the shorten-operation is never executed after run r. Therefore, each run either ends with a remove-operation or it shifts a vector to the end of the configuration.

Assume now a run $r' \geq r$ with $C_{r'} = (\vec{u}_0, \ldots, \vec{u}_k)$. If $|\vec{u}_0| < \frac{1}{2}\mathcal{R}$ and $|\vec{u}_i| < \frac{1}{2}\mathcal{R}$ and no remove-operation has been executed by the run prior to the movement of v_{i+1}, then we have by our invariant $p'(i) = p(i+1) - \vec{u}_0$.

By the triangle inequality the distance between $p'(i)$ and $p(i+2)$ is smaller than $\mathcal{R}$, as $\vec{u}_0 < \frac{1}{2}\mathcal{R}$ and $\vec{u}_i < \frac{1}{2}\mathcal{R}$. Then, the relay v_{i+1} executes the remove-operation.

On the contrary, if $|\vec{u}_0| \geq \frac{1}{2}\mathcal{R}$ and $|\vec{u}_i| \geq \frac{1}{2}\mathcal{R}$, then the relay v_{i+1} cannot execute the remove-operation, as by $\angle(\vec{u}_0, \vec{u}_i) > \pi/2$ the distance $\left|p'(i) - p(i+2)\right|_2 > \mathcal{R}$.

Denote by S_r the longest suffix of the configuration containing only vectors with length larger than $\frac{1}{2}\mathcal{R}$. Assume now that before run r the vector $\vec{u}_j$ is the first vector in S_r and that there is more than one vector with size smaller than $\frac{1}{2}\mathcal{R}$ in the configuration. If $|\vec{u}_0| < \frac{1}{2}\mathcal{R}$ then the remove-operation will be executed either by relay v_i, such that $|\vec{u}_{i-1}| < \frac{1}{2}\mathcal{R}$ or by an earlier relay, so that S_r is left untouched. If $|\vec{u}_0| \geq \frac{1}{2}\mathcal{R}$ then by our earlier observation the remove-operation cannot influence S_r as all vectors in S_r have length at least $\frac{1}{2}\mathcal{R}$.

Therefore, after at most k shifts, there is at most one vector with size $\frac{1}{2}\mathcal{R}$. As there may be at most k remove-operations, there are k shifts in $2k$ runs. ■

Let C be the configuration after $8n+1$ runs. Using the former lemma, we can conclude that there are at most $2|C|+1$ relays in the chain after this time. With the bound on $|C|$ from Lemma 13, part (c) of Theorem 6 follows, concluding the proof of this theorem.

4.2.2. Performance in the Dynamic Scenario

The basic idea used for transforming the *Hopper* strategy into *Dynamic-Hopper* is very similar to that used for the *Dynamic-Manhattan-Hopper* strategy. The *Dynamic-Hopper* strategy is executed in rounds, such that at the beginning of the round the explorer moves, then a follow-run and a series of α hopper-runs are executed. The constant α will be defined later. We assume that all runs are pipelined, as shown on Fig. 4.6. Whenever we will be talking of runs, we mean hopper-runs.

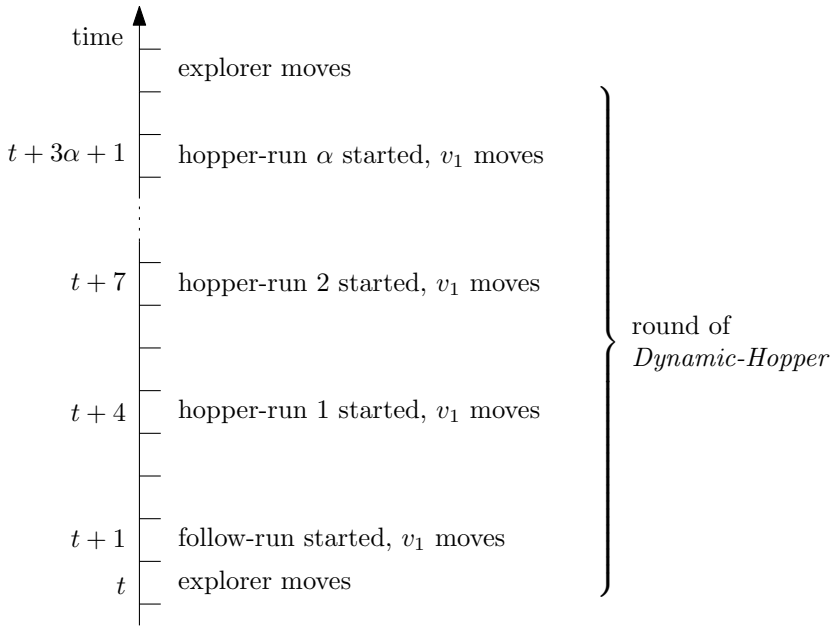


Figure 4.6.: Round of *Dynamic-Hopper* strategy.

We assume that each movement of the explorer has a distance of at least $\frac{1}{2}\mathcal{R}$ and at most 1. Furthermore, for the sake of technical simplicity we assume all vectors in the configuration in the first time step have length at least $\frac{1}{2}\mathcal{R}$. Then we are able to show the following theorem.

Theorem 4.15. *There exists a constant α such that for the chain maintained by the Dynamic-Hopper strategy the following holds.*

(a) *The chain remains connected,*

(b) *the explorer may move a distance of $\frac{1}{2}\mathcal{R}$ every round, i.e. every $3(\alpha + 1)$ time steps,*

(c) *relays move at most distance $\alpha\sqrt{2} \cdot \mathcal{R}$ per round,*

(d) *the number of relays used in the chain is $O(d_r/\mathcal{R})$ at the beginning of round r.*

Parts (a), (b), and (c) are obvious from the definition of the strategy. In the rest of the chapter we will prove part (d), thereby specifying the constant α.

Virtual length. We can define the virtual length of a spatial vector to be

$$\|\vec{u}\| = \begin{cases} |\vec{u}| & \text{if } |\vec{u}| \geq \frac{1}{2}\mathcal{R} \\ |\vec{u}| + \frac{1}{2}\mathcal{R} & \text{otherwise} \end{cases}$$

This notion has the advantage that a vector has always virtual length at least $\frac{1}{2}\mathcal{R}$. The virtual length of a configuration is obviously the sum of the virtual lengths of all vectors belonging to this configuration. In contrast, the length $|\vec{u}|$ will be called the real length.

We want to investigate how the operations performed by the hopper-runs influence the virtual length. For the remove-operation, note that the sum of the participating vectors' virtual length is at least $\mathcal{R}$. On the other hand, the vector created by the remove-operation has virtual length at most $\mathcal{R}$. Therefore, the remove-operation does not increase the virtual length of the chain.

For the shorten-operation, either both of the participating vectors have real length at least $\frac{1}{2}\mathcal{R}$ and then by Lemma 4.9 the operation reduces the virtual chain length by at least $\frac{1}{3}\mathcal{R}$. Now assume that one of the vectors $\vec{u}_0$ and $\vec{u}_1$ participating in the shorten-operation has real length smaller than $\frac{1}{2}\mathcal{R}$. W.l.o.g. let it be $\vec{u}_0$. For the real length of the resulting vector it holds $|\vec{u}_0 + \vec{u}_1| \leq |\vec{u}_0| + |\vec{u}_1|$. By the definition of the virtual length we have $|\vec{u}_0| + |\vec{u}_1| \leq \|\vec{u}_0\| - \frac{1}{2}\mathcal{R} + |\vec{u}_1|$. This implies that the shorten operation reduces the virtual length of the chain by at least $\frac{1}{2}\mathcal{R} > \frac{1}{3}\mathcal{R}$.

Fact 4.16. *Every shorten-operation reduces the virtual length of the chain by at least* $\frac{1}{3}\mathcal{R}$.

As each relay has virtual length at least $\frac{1}{2}\mathcal{R}$, the virtual length can be used to bound the number of relays in the chain.

Fact 4.17. *Let l be the virtual length of the chain. Then the number of relays is bounded from above by* $2l$.

Phases. We will divide the runtime of the *Dynamic-Hopper* strategy in phases. The first phase starts in the first round. Let n denote the number of relays in the chain, l be its virtual length and d the distance between the base camp and the explorer in the beginning of the phase. Similarly, n', l' and d' denote the same at the end of the phase.

Throughout the analysis we use three constants. By α we denote the number of hopper-runs in a round, γn is the number of rounds in a phase. Furthermore, we assume that at the beginning of the phase $l \leq \beta d$ holds.

We require $\beta \geq 2\sqrt{2}+1$ and the constants γ and α to fulfill

$$\gamma \leq \frac{1}{2\sqrt{2}(10\beta+14)},$$

and $\alpha \geq \frac{8}{\gamma}+8$.

Infected vectors. At the beginning of a phase all vectors are marked as non-infected. Every new vector introduced by the movement of the explorer to the configuration is called infected. Every time a remove-operation or a shorten-operation removes at least one infected vector, the vectors created by the operation are marked as infected. Intuitively, the marking of infected vectors allows us to distinguish vectors which have been influenced by the movement of the explorer from the current phase.

A non-infected vector is said to be non-conflicting if it has no conflict to any non-infected vectors.

Outline of the analysis. We will first show that at the end of the phase the non-infected vectors have real length at least $\frac{1}{2}\mathcal{R}$ and that they have all angles larger than $\pi/2$ with respect to each other. Afterward, we will distinguish two cases: either at the end of the phase there are more than $(3\beta+4)\gamma n$ infected vectors or not. In the first case we will show that so many infected vectors could have been only created by many shorten-operations which also decreased the length of the chain significantly. In the second case we will be able to show that thanks to the majority of vectors being non-infected the chain length is linear in the distance d_r. In both cases we will show that $l' \leq \beta d'$.

Lemma 4.18. *At the end of a phase all non-infected vectors are non-conflicting and all but at most one have real length at least* $\frac{1}{2}\mathcal{R}$.

Proof. Let us follow the changes in the position of a non-infected vector in configurations. For that purpose, whenever a vector is replaced by the shorten-operation we assign its identification to the new vector created. In case of the remove-operation we assign the new vector the identification of one of the removed vectors.

Whenever a vector is not at the first position in the configuration, it's position can change in the following manner

1. decrease by 1 if the run shifts a vector to the end of the configuration,
2. decrease by 1 or without changes in case of a remove-operation or shorten-operation,

3. increase by 1 if the explorer has moved and a new vectors has been added.

As there are at most $3(n+\gamma n)+(n+\gamma n)$ runs ending in Case 2 and at most $n+\gamma n$ occurrences of Case 3, after $7(n+\gamma n)$ runs each non-infected vector has been at the first position of the configuration and has been either shifted as non-conflicting to the end, or removed. Therefore, after $7(n+\gamma n)$ runs all non-infected vectors are non-conflicting. After at most $8(n+\gamma n)$ runs each vector has been on the first position for at least two times, and then following the argumentation from Lemma 4.14 all but one vectors have real length at least $\frac{1}{2}\mathcal{R}$.

By the definition of α, we have $\alpha\gamma n \geq 8(n+\gamma n)$ and the lemma follows. ■

Lemma 4.19. *Let there be at least* $(3\beta+4)\gamma n$ *infected vectors at the end of the phase. Then* $l' \leq \beta d'/\mathcal{R}$.

Proof. The number of infected vectors in a phase generated by the movement of the explorer is at most γn. Note that only the shorten-operation is able to generate two new infected vectors out of one, thereby increasing the number of infected vectors in the configuration. Therefore, in order to have $(3\beta+4)\gamma n$ infected vectors at the end of the phase, at least

$$(3\beta+4)\gamma n - \gamma n \geq 3(\beta+1)\gamma n$$

shorten-operations must have been performed during this phase.

Note that each of the shorten-operations decreases the virtual length of the chain by at least $\frac{1}{3}\mathcal{R}$ and the only operations enlarging the chain are the movements of the explorer. So, we have $l' \leq l-(\beta+1)\gamma n\cdot\mathcal{R}+\gamma n\cdot\mathcal{R} = l-\beta\gamma n\cdot\mathcal{R}$. Since the explorer decreased its distance to the base camp by at most γn it follows that $d' \geq d-\gamma n\cdot\mathcal{R}$. From the definition of γ it follows $\gamma < \frac{1}{2\beta}$ which, thanks to $n \leq \frac{2}{\mathcal{R}}l \leq \frac{2}{\mathcal{R}}\beta d$, implies $d-\gamma n\cdot\mathcal{R} > 0$. We obtain

$$\frac{l'}{d'} \leq \frac{l-\beta\gamma n\cdot\mathcal{R}}{d-\gamma n\cdot\mathcal{R}} \leq \frac{l}{d} \leq \beta\,.$$

The last inequality holds because $l \leq \beta d$. This implies the desired bound on l'. ■

Lemma 4.20. *Let there be at most* $(3\beta+4)\gamma n$ *infected vectors at the end of the phase. Then* $l' \leq \beta d'$.

Proof. Sum up the non-infected vectors to a vector $\vec{u}$ and the infected vectors to a vector $\vec{a}$. Obviously, $p'(0)+\vec{u}+\vec{a} = p'(k+1)$ where v_{k+1} is the base camp. Define the

point $\rho = p'(0) + \vec{u}$. Since all non-infected vectors have angle larger to $\pi/2$ with respect to each other, by Lemma 4.13 the actual length of the non-infected vectors is at most $\sqrt{2}\left|\rho - p(0)\right|_2$. Furthermore, by Lemma 4.18 all but one non-infected vectors have real length equal to virtual length. Thus the virtual length of the non-infected vectors L' is bounded by $L' \leq \sqrt{2}\left|\rho - p(0)\right|_2 + \frac{1}{2}\mathcal{R}$. The virtual length of the chain is bounded by $l' \leq L' + |\vec{a}|$ and since $|\vec{a}| \leq (3\beta + 4)\gamma n \cdot \mathcal{R}$ we have $L' \geq l' - (3\beta + 4)\gamma n \cdot \mathcal{R}$.

Therefore,

$$\begin{aligned} d' &= \left|p'(0) - p'(k+1)\right|_2 \geq |\vec{u}| - |\vec{a}| \geq \tfrac{1}{\sqrt{2}} L' - \tfrac{1}{2\sqrt{2}}\mathcal{R} - (3\beta + 4)\gamma n \cdot \mathcal{R} \\ &\geq \tfrac{1}{\sqrt{2}} l' - (5\beta + 7)\gamma n\mathcal{R} - \tfrac{1}{2\sqrt{2}}\mathcal{R}\,. \end{aligned} \tag{4.1}$$

As used earlier we have $n \leq 2l/\mathcal{R} \leq 2\beta d/\mathcal{R}$. On the other hand $d' \geq d - \gamma n \cdot \mathcal{R}$. Combining both bounds we obtain

$$n \leq d' \frac{2\beta}{\mathcal{R}(1 - 2\beta\gamma)}\,.$$

Plugging in the bound on n into Eq. (4.1) and assuming $d' \geq \mathcal{R}$ we obtain

$$\frac{l'}{d'} \leq \sqrt{2}\left(1 + \tfrac{1}{2\sqrt{2}} + \frac{2\beta\gamma(5\beta + 7)}{1 - 2\beta\gamma}\right) \leq \beta\,,$$

as $\gamma \leq \frac{1}{2\sqrt{2}(10\beta+14)}$ is sufficiently small. ■

Applying both Lemma 4.19 and Lemma 4.20, we have $l \leq \beta d$ at the beginning of each phase. During the phase the number of vectors may increase by at most γn and the distance between the explorer and the base camp may decrease by at most $\gamma n \cdot \mathcal{R}$. Denote by n_t the number of relays in the chain in time step n_t and the distance between explorer and base camp by d_t. Then we have

$$\frac{n_t}{d_t} \leq \frac{n + n\gamma}{d - n\gamma \cdot \mathcal{R}} \leq \frac{2\beta(1 + \gamma)}{1 - 2\beta\gamma}\,,$$

as $d \geq \frac{n}{2\beta \cdot \mathcal{R}}$. For the defined value of γ we have $\frac{2\beta(1+\gamma)}{1-2\beta\gamma} \leq \beta^2$ and therefore the number of relays in the chain at any point of time is at most $\beta^2 d_t/\mathcal{R}$. This concludes the proof of part (d) of Theorem 16.

CHAPTER 5

Chase-Explorer Strategy

As the last of the strategies used for organizing a chain of relays we introduce the *Chase-Explorer* strategy. The greatest difference between the *Chase-Explorer* strategy and the two methods introduced earlier is that the relays now require some information about the position of the base camp, relatively to their current position. This implies, that the relays have to be stateful in order to save the position of the base camp.

This model corresponds to a setting where relays are equipped with GPS-like devices which determine their global position with a bounded error.

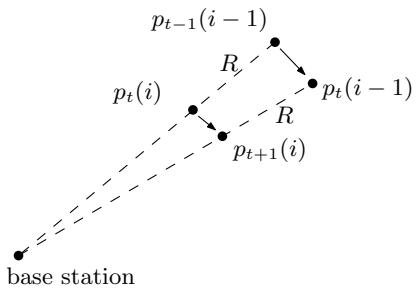


Figure 5.1.: The *Chase-Explorer* strategy

Strategy Description

In the *Chase-Explorer* strategy all relays work in parallel as in *Go-To-The-Middle*. At the beginning of a time step t, relay v_i looks at the position of its predecessor $p_t(i-1)$. Then it computes the coordinates of a point which is on the line segment connecting $p_t(i-1)$ and the base camp's position b, and is in distance $Z := \mathcal{R} - 1$ from $p_t(i-1)$. The positions are depicted in Fig. 5.1. Informally speaking, the relays try to keep as near to the direct line connecting the base camp and the explorer as possible. At the same time they try

to maintain a distance of Z to the previous station in the chain. This is different to *Go-To-The-Middle*, where stations place themselves according to the positions of their neighbors only.

In the dynamic scenario, the movement of the explorer can change its distance to the base camp. Thus, it may be necessary to change the number of relays in the chain. The base camp decides to insert a new relay station when the last relay v_{n_t} reaches a distance larger than Z to the base camp at the end of time step $t-1$. The new relay is inserted on the line between the position $p_t(n_t)$ and the base camp, keeping a distance of Z to the position $p_t(n_t)$. On the contrary, if the next to last relay comes too near to the base camp rendering the last relay useless, the last one is removed.

Outline

We start our analysis of the *Chase-Explorer* strategy with proving its correctness. In Section 5.2 we will show the performance of the *Chase-Explorer* strategy in the static scenario and in Section 5.3

Afterward, in Section 5.4 we will analyze the behavior of the chain in presence of imprecise localization methods, where stations only know an approximation of the position of the base camp, with an appropriately bounded additive error.

5.1. Correctness

To ensure correctness of the *Chase-Explorer* strategy it is necessary to show that the relays will be able to chase the explorer without exceeding their maximum speed and that the distance between neighbored stations won't exceed $\mathcal{R}$.

We first investigate the correctness of the *Chase-Explorer* strategy assuming no localization errors.

Theorem 5.1. *Assuming the explorer moves with a speed of at most* 1 *per time step, it holds*

- *the speed of the relays does not exceed* 1 *per time step,*
- *the distance between neighbors in the chain never exceeds* $\mathcal{R}$.

Proof. First we want to show that no relay is required to move for a distance larger than 1 during one time step, providing that the explorer does not exceed its maximal speed of 1 per time step. Assume that station v_{i-1} moves a distance of at most 1 every time step. A movement of station v_{i-1} is depicted on Fig. 5.1 between points $p_{t-1}(i-1)$ and

$p_t(i-1)$. The distances $|p_t(i) - p_{t-1}(i-1)|_2 = |p_{t+1}(i) - p_t(i-1)|_2$ are both equal to Z. By an obvious geometric argument the distance between $p_t(i)$ and $p_{t+1}(i)$ traveled by station v_i in time step t is not greater than the distance between $p_{t-1}(i-1)$ and $p_t(i-1)$. So, the movement distance of station v_i is bounded by 1 if station v_{i-1} has moved by at most 1.

We consider the distance between a station v_i and its neighbor v_{i-1} at the end of time step t. Obviously the distance between $p_t(i)$ and $p_{t-1}(i-1)$ is exactly Z after time step t. Since the station v_{i-1} can move for a distance of at most 1 during the time step t, the distance between $p_{t-1}(i-1)$ and $p_t(i-1)$ is at most 1. Thus, by the triangle inequality, the distance between $p_t(i)$ and $p_t(i-1)$ is at most $Z+1$. This assures that the distance between the relay v_i and the explorer is at most $Z + 1 \leq \mathcal{R}$ at the beginning of each time step. ■

5.2. Performance in Static Scenario

The *Chase-Explorer* strategy as it has been defined earlier is naturally designed for the dynamic scenario. In the dynamic scenario we have been maintaining the invariant that a relay is able to move to its computed position during one time step. We won't be able to keep up this assumption in the static scenario, with a chain configuration given at the beginning which is far from the optimal. Nevertheless we have to assume that the distance between neighbored relays at the beginning of the first time step does not exceed Z, otherwise the chain could become disconnected after the first time step.

In the modified strategy, a relay v_i first computes the intersection of a circle with radius 1 around $p_t(i)$ and another circle with radius Z around $p_t(i-1)$. Obviously, every point in this intersection is in distance at most Z from $p_t(i-1)$ and in distance at most 1 from $p_t(i)$. Therefore we bound all relays to move to one of the points from this intersection. It can be shown by a straightforward inductive argument similar to that in Theorem 5.1 that in this case the distance between neighbored relays never exceeds $\mathcal{R}$ and that the movement distance does not exceed 1, assuming that the distance between relays at the beginning of the first time step does not exceed Z.

The relays choose the point from the intersection which is nearest to the point on the line segment between the predecessor and the base camp, which was computed in the original *Chase-Explorer* strategy. For the modified strategy we are able to show the following theorem.

Theorem 5.2. *In the static scenario the Chase-Explorer strategy reduces the chain length to optimal in $2Z \cdot n$ time steps, where n is the number of relays initial chain. Finally, the chain uses at most $\left\lfloor d\frac{1}{\mathcal{R}-1} \right\rfloor$ relays.*

Proof. Assume that the station v_{i-1} remains immobile, so that relay v_i does not have to follow v_{i-1} but can concentrate on aligning itself properly with respect to v_{i-1}. Let ρ designate optimal position for the relay v_i, i.e. the point on the line segment between $p_t(i-1)$ and the base camp, in distance Z to $p_t(i-1)$.

As the distance between $p_t(i-1)$ and $p_t(i)$ is at most Z, the line segment between $p_t(i)$ and ρ is completely inside of the circle with radius Z around $p_t(i-1)$. Therefore, the relay v_i can decrease its own distance to ρ by 1 in each time step. Thus, in at most $2Z$ time steps the relay is able to reach ρ.

In the static scenario the explorer v_0 obviously does not move, so that v_1 is able to reach its optimal position after $2Z$ time steps. Inductively we have that relay v_i does not move after $2Z \cdot i$ time steps and that therefore the chain has optimal length after at most $2Z \cdot n$ time steps, where n is the number of relays at the beginning. ■

5.3. Performance in Dynamic Scenario

Due to the nature of the *Chase-Explorer* strategy, the explorer is always able to move freely with a speed of 1 per time step, without being restricted by the relays.

Theorem 5.3. *The number of stations in the chain at any point of time is at most* $\left\lfloor \frac{1}{\mathcal{R}-2} d_t \right\rfloor$ *times more than in the optimal chain.*

Proof. Observe a pair of stations, v_i and v_{i-1} at the beginning time step t with $i < n_t + 1$. As $p_t(i)$ is on the line segment between $p_{t-1}(i-1)$ and the base camp, we have $\left|b - p_t(i)\right|_2 = \left|b - p_{t-1}(i-1)\right|_2 - Z$. As v_{i-1} might have moved only by at most 1 in time step $t+1$, we have $|b - p_t(i)|_2 \leq |b - p_t(i-1)|_2 - Z - 1$. Therefore, relay v_i is nearer to the base camp than v_{i-1} by at least $Z - 1$. This yields an upper bound of $\left\lfloor \frac{1}{\mathcal{R}-2} d_t \right\rfloor$ on the number of relays in the chain. ■

5.4. Imprecise Base Camp Localization

In the following, we will consider the behavior of the *Chase-Explorer* strategy in presence of imprecise estimates of the base camp's position. Formally, we denote the position of the base camp known by station v_i as $b_t(i)$ at time step t, whereas b is the real base camp's position.

It is not hard to imagine that the strategy is able to work with a localization scheme with some bounded additive error ϵ (i.e. the GPS, so that $|b - b_t(i)|_2 \leq \epsilon$). We though want to show that a weaker localization scheme is enough for *Chase-Explorer* to work properly and to attain a reasonable performance (in terms of the number of relays used in the chain). We define the hop distance of relay v_i to the base camp to be $\gamma_t(i) := n_t - i + 1$. We aim at showing that a localization system such that $|b_t(i) - b|_2 \leq \epsilon \cdot Z \cdot \gamma_t(i)$ is enough. This means, that stations which are further apart from the base camp are allowed to have a larger error in their localization. Note that the localization error causes relays to position next to the direct line between their predecessor and the base camp, increasing the number of relays in the chain.

This weak accuracy brings some problems: since the accuracy depends on the number $\gamma_t(i)$ and the number $\gamma_t(i)$ depends on the accuracy one might be worried that the chain gets infinitely long, with the accuracy getting weaker parallely. Theorem 5.4 shows that this behavior does not occur if ϵ is bounded sufficiently. Theorem 5.8 shows the contrary, i.e. that a localization system with ϵ large can lead to unstable behavior.

Theorem 5.4. *If* $|b_t(i) - b|_2 \leq \frac{Z}{25} \cdot \gamma_t(i)$ *then* $n_t \leq 1.5 \cdot d_t \frac{1}{Z} + 1$.

Proof. Let $r_t(i) := |b - p_t(i)|_2$ be the distance of the station v_i to the base camp in time step t. Then we define

$$\begin{aligned} u_t(i) &:= r_{t-1}(i) - r_t(i+1) \\ u_t(i) - 1 \leq \tilde{u}_t(i) &:= r_t(i) - r_t(i+1) \end{aligned}$$

Let $\alpha_{t-1}(i)$ be the angle at $p_{t-1}(i)$, between the line segments $\langle b, p_{t-1}(i)\rangle$ and $\langle b_t(i+1), p_{t-1}(i)\rangle$, as shown on Fig. 5.2. With other words $\alpha_{t-1}(i)$ is the angle, measured at $p_t(i)$ between the real position of the base camp and the approximation of the base camp's position known by v_{i+1}.

Let $|b - b_t(i+1)|_2 \leq \epsilon_t(i+1) = \epsilon \cdot Z \cdot \gamma_t(i)$. We introduce three lemmas, which relate the values of $u_t(i), \alpha_{t-1}(i), r_{t-1}(i)$ to each other

Lemma 5.5. *For every i and t such that* $\gamma_t(i) > 2$

$$u_t(i) \geq Z \cdot (\cos(\alpha_{t-1}(i)) - \sin(\alpha_{t-1}(i))) .$$

Proof. In order to simplify notation let us set $r := r_t(i+1)$, $u := u_t(i)$, $\alpha := \alpha_{t-1}(i)$. From the Law of Cosines we have (see Fig. 5.2)

$$r^2 = (r+u)^2 + Z^2 - 2(r+u) \cdot Z \cdot \cos\alpha .$$

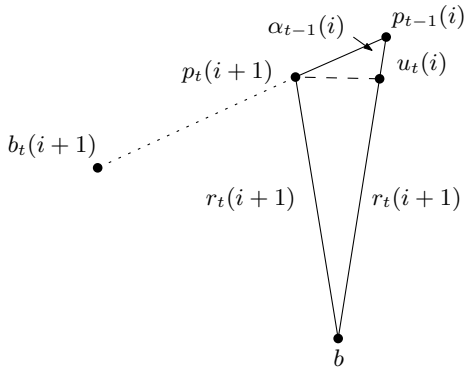


Figure 5.2.: Value $u_t(i)$ and $\alpha_{t-1}(i)$

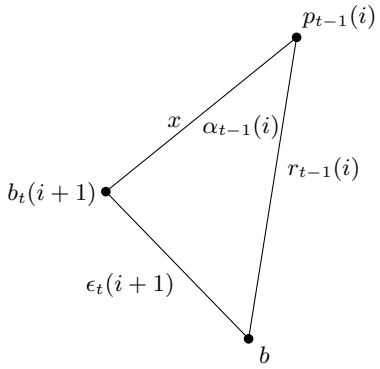


Figure 5.3.: Imprecise localization of the base camp

Looking at the above as on a polynomial of u yields the following positive root

$$u = Z \cdot \cos\alpha + \sqrt{r^2 - Z^2 \sin^2\alpha} - r \geq Z \cdot (\cos\alpha - \sin\alpha) .$$

The last inequality comes from the fact that $\sqrt{r^2 - Z^2 \sin^2\alpha} - r$ is a non-decreasing function of r. Since $r \geq Z \cdot \sin\alpha$ for the triangle to exist, we can plug in $r = Z \cdot \sin\alpha$ and obtain the requested lower bound. ■

Lemma 5.6. *For every* $\alpha_{t-1}(i)$

$$\begin{aligned} \cos\alpha_{t-1}(i) &\geq \sqrt{1 - \frac{\epsilon_t^2(i+1)}{r_{t-1}^2(i)}} , \\ \sin\alpha_{t-1}(i) &\leq \frac{\epsilon_t(i+1)}{r_{t-1}(i) - \epsilon_t(i+1)} . \end{aligned}$$

Proof. Let us consider the triangle between b, $b_t(i+1)$ and $p_t(i)$ as shown on Fig. 5.3. Stating $\cos\alpha$ as a function of $x = \left|b_t(i+1) - p_{t-1}(i)\right|_2$ one obtains

$$\cos\alpha = \frac{x^2 + r_{t-1}(i)^2 - \epsilon_t(i+1)^2}{2x \cdot r_{t-1}(i)} \,. \tag{5.1}$$

This function obtains its minimum for

$$x = \sqrt{r_{t-1}(i)^2 - \epsilon_t(i+1)^2} \,. \tag{5.2}$$

Plugging in Eq. (5.2) into Eq. (5.1) yields the lower bound on $\cos\alpha$. Using the same triangle one obtains the upper bound on

$$\sin\alpha_{t-1}(i) = \frac{\epsilon_t(i+1)}{x} \leq \frac{\epsilon_t(i+1)}{r_{t-1}(i) - \epsilon_t(i+1)} \,.$$

■

Lemma 5.7. *Let $u \leq u_{t-1}(i)$ and $u \geq 3$ for all t and i such that $i \leq n_t - 2$. Then for all t and i such that $i \leq n_t - 2$ it holds*

$$r_{t-1}(i) \geq \frac{1}{3} \cdot \gamma_t(i+1) \cdot u$$

Proof. Observe that the distance $r_{t-1}(i)$ is equal to a sum of $\tilde{u}_{t-1}$

$$r_{t-1}(i) = \sum_{j=i}^{n_{t-1}} \left(r_{t-1}(j) - r_{t-1}(j+1)\right) = \sum_{j=i}^{n_{t-1}} \tilde{u}_{t-1}(j) \,.$$

Observe that $n_{t-1} \geq n_t - 1$, since the number of relay stations in the chain can change by at most 1 during one time step. Then we have

$$\begin{aligned} r_{t-1}(i) &\geq \sum_{j=i}^{n_{t-1}} (u_{t-1}(j) - 1) \\ &\geq u_{t-1}(n_{t-1}) + (n_{t-1} - i) \cdot \min_{j=i,\ldots,n_{t-1}-1} u_{t-1}(j) - 1 \\ &\geq (n_t - i - 1) \cdot (u - 1) \,. \end{aligned}$$

Note that we have lower bounded $u_{t-1}(n_{t-1})$ by 0, since the last relay station can be very near to the base station. Since $u \geq 3$ it follows

$$r_{t-1}(i) \geq (n_t - i - 1)(u - 1) \geq \frac{1}{3} \cdot \gamma_t(i+1) \cdot u$$

■

Take any time step $t-1$. Let $u = 0.7 \cdot Z$. We want to show that $u_t(i) \geq u$ for all i such that $\gamma_t(i) \geq 1$ if $u \leq u_{t-1}(i)$ and $Z \geq 5$. For the station standing next to the base camp we assume that it may be at any distance to the base camp, therefore $u_t(n_t - 1)$ may be as low as 0.

Bringing together Lemma 5.5, 5.6, and 5.7 we obtain the following lower bound

$$u_t(i) \geq Z \cdot \left(\sqrt{1 - \frac{9 \cdot \epsilon^2 \cdot Z^2}{u^2}} - \frac{\epsilon \cdot Z}{1/3 \cdot u - \epsilon \cdot Z} \right).$$

To prove our claim it should hold

$$Z \cdot \left(\sqrt{1 - \frac{9 \cdot \epsilon^2 \cdot Z^2}{u^2}} - \frac{\epsilon \cdot Z}{1/3 \cdot u - \epsilon \cdot Z} \right) \geq u \,.$$

Plugging in $u = 0.7 \cdot Z$ the above holds for all $\epsilon \leq 1/25$. Therefore we have $u_t(i) \geq 0.7 \cdot Z$ for all i and t such that $\gamma_t(i) \geq 1$.

For the sake of contradiction assume now that there is a time step such that the number of relays used by *Chase-Explorer* exceeds $1.5\lceil d_t/Z \rceil + 1$. By our previous considerations this would mean that $d_t \geq 0.7 \cdot Z\,(1.5\lceil d_t/Z \rceil + 1) \geq 1.05 \cdot d_t > d_t$, which is clearly a contradiction. ■

For the next theorem we assume that the may dictate the values $b_t(i)$ as long as $|b - b_t(i)| \leq \epsilon \mathcal{R} \gamma_t(i)$.

Theorem 5.8. *If the adversary can dictate errors with $\epsilon \geq 1$, then the number of relays in the chain is unbounded.*

Proof. The distance of relay v_i to the base camp clearly cannot exceed $Z \cdot \gamma_t(i)$. So, if we have $\epsilon \geq 1$ the adversary can select a position $b_t(i)$ all around station v_i. This allows the adversary to completely control the shape of the chain. More specifically, it can create a chain such that $r_t(i) \geq r_t(i-1)$. This part of the chain increases its distance to the relays instead decreasing it. ■

Part II.

Managing Multiple Communication Chains

CHAPTER 6

Minimizing the Number of Changes in Support Graph

In this chapter we introduce the Online Dynamic Minimum Spanning Tree (ODMST) problem. We are looking for a minimum spanning tree (MST) in a graph, whose edge weights change dynamically. The problem is online in that the changes in the graph cannot be foreseen and we have to deal with them as they arrive. Therefore, we will investigate algorithms solving the ODMST problem in terms of competitive analysis.

An online algorithm solving the ODMST problem maintains a MST throughout time. This means, that every time the weight of an edge is changed, the online algorithm must output a minimum spanning tree for the new graph. If possible, this spanning tree should be the same spanning tree as computed in the previous time step. This is not always possible, as there may be a new tree with smaller weight. In this case, the algorithm should output a MST which is as similar to its last tree as possible. The cost of the algorithm in a time step is equal to the number of differences in terms of edges between the current MST and the previous one. Note, that the cost model does not take the computational effort, which is necessary for computing a new MST into account; we are only concerned with the number of changes in the MST here.

The MST maintained by an algorithm solving the ODMST problem can be used as the support graph for a compacted robot graph. Thereby the support graph has the minimum possible length among all connect support graphs. This implies that the number of relays used for establishing the communication chains is minimal. The greatest advantage of the support graph defined in this way is that it changes only seldom. As each change in the support graph requires relays to move from one path in terrain to another, we are hereby minimizing the energy used by the relays to move.

We continue now with a brief discussion on competitive analysis, followed by a formal model of the ODMST problem. After providing the reader with basic notation

we will be able to sketch the results presented in this chapter for the ODMST problem. The introductory part concludes with a discussion on research related to the ODMST problem and the deficiencies of the model presented herein.

Competitive Analysis

The ODMST problem and the problem investigated in Chapter 7 are analyzed in the common framework of competitive analysis. This methodology allows to study algorithms which work as long-running processes. As such processes, they have to act basing on the knowledge they have obtained so far, without knowing the details of the situations they will encounter in the future. One might imagine that this may easily lead online algorithms to decisions, which they wouldn't have made when knowing the future (e.g. if you do not know the future lottery numbers, you probably won't choose the right ones).

From the formal point of view, online algorithms are presented an *input sequence* σ. Usually one assumes that the time is divided into discrete time steps and only one item of the input sequence is revealed to the online algorithm per time step. The online algorithm is forced to perform an action corresponding to the item of the input sequence right after receiving it. We will generally assume that each input sequence item is some kind of request, which must be served by the online algorithm right after it is received. Handling this request causes some cost to the algorithm. The cost for serving a particular request depends very much on how the algorithm has behaved in the past. The cost of the algorithm ALG for the whole input sequence is the sum of its costs over all time steps and is denoted by $C_{\mathrm{ALG}}(\sigma)$.

For the purpose of competitive analysis one compares the cost of this online algorithm to the cost of an optimal offline algorithm – this optimal offline algorithm has access to the whole input sequence from the beginning on, and can therefore make optimal decisions. This allows it to minimize the total cost of serving the sequence of requests. We denote the cost of the optimal algorithm for sequence σ by $C_{\mathrm{OPT}}(\sigma)$. The central performance measure employed in competitive analysis is the competitive ratio. We say that the online algorithm has a competitive ratio $\mathcal{R}$ if

$$C_{\mathrm{ALG}}(\sigma) \leq \mathcal{R} \cdot C_{\mathrm{OPT}}(\sigma) + c,$$

for all feasible input sequences $\sigma \in \Sigma$ and a proper constant c independent of the input sequence. The goal of the designer of an online algorithm is to minimize the ratio $\mathcal{R}$.

For convenience we will often assume there is some adversary, which constructs the input sequence σ, trying to maximize the ratio $C_{\mathrm{ALG}}(\sigma)/C_{\mathrm{OPT}}(\sigma)$. For this purpose she[1]

[1] For some reason unknown to us the adversary is traditionally found to be female in literature. For the sake of compliance, although with a slight discomfort, we will stick to this notation.

has to construct this sequence so that it incurs a cost as high as possible to the online algorithm while keeping the cost of the optimal offline algorithm low. It is assumed that the adversary knows the complete description of the online algorithm (i.e. its source-code, machine-code). Therefore in case of a deterministic online algorithm it can simulate the algorithm's behavior in advance and construct an input sequence σ which maximizes the ratio $C_{\mathrm{ALG}}(\sigma)/C_{\mathrm{OPT}}(\sigma)$.

Since we want also to investigate the performance of randomized online algorithms, we have to refine our notion of adversary. We still assume that the adversary knows the description of the online algorithm, on the other hand it does not know the outcome of the random experiments performed by the online algorithm. She has to prepare the input sequence before the online algorithm is run on it. This type of adversary is known as an *oblivious* adversary. We can redefine the notion of competitive ratio $\mathcal{R}$ to the expected competitive ratio such that

$$\mathbf{E}_{\phi}\left[C_{\mathrm{ALG}}(\sigma)\right] \leq \mathcal{R} \cdot C_{\mathrm{OPT}}(\sigma) + c,$$

where the expected value is taken with respect to the random choices ϕ of the online algorithm. Constructing an input sequence very complicated for the online algorithm is not as easy for the adversary as in the case of deterministic algorithms – she does not know the algorithm's random decisions and therefore does not know how exactly the algorithm serves its requests. We will see that in this framework one is able to design algorithms with better competitive ratios than is possible for deterministic algorithms.

Stochastic adversaries The described competitive ratio notions are worst-case notions, i.e. we are maximizing over all possible input sequences. In many situations this may appear too pessimistic as a notion of performance for algorithms. A weaker notion of adversaries is represented by stochastic adversaries, introduced by [BKR+01, Bie05], which create the input sequence by means of a stochastic process. The stochastic process chooses the input sequence according to some probability distribution π. We say that a deterministic algorithm ALG has a competitive ratio R with a probability of $1-\gamma$ if

$$\mathbf{Pr}\left[C_{\mathrm{ALG}}(\sigma) \leq R \cdot C_{\mathrm{OPT}}(\sigma) + c\right] \geq 1-\gamma\ ,$$

where the constant c does not depend on σ. The described probability is calculated with respect to the probability distribution π over all input sequences.

Formal Model of the ODMST Problem

Let $G = (V, E)$ be a graph with edges weighted initially by the function $w^0 : E \to \mathbb{N}_+$. We denote $m = |E|$. Time is divided into discrete time steps. The input sequence σ defines

the changes in weights of edges. In time step i we have $\sigma(i) \in E \times \{-1, +1\}$, thereby only one edge changes its weight in one time step and the change is bounded to be either $+1$ or -1.

Basing on the original input sequence, for convenience of notation, we denote by $\delta(i, e) \in \{-1, 0, 1\}$ the change of weight of edge $e \in E$ in the i-th time step. Formally we have

$$\delta(i, e) = \begin{cases} -1 & , \text{ if } \sigma(i) = (e, -1) \\ 1 & , \text{ if } \sigma(i) = (e, +1) \\ 0 & , \text{ otherwise .} \end{cases}$$

Furthermore, we introduce the function $w : \mathbb{N}_+ \times E \to \mathbb{N}_+$ which maps a time step t and edge e to the edge weight at the beginning of time step t. This gives

$$w^t(e) = w^0(e) + \sum_{i=1}^{t} \delta(i, e) .$$

Note that hereby $w^t(e)$ defines the weight of edge e including the change happening in the t-th time step.

An algorithm ALG solving the Online Dynamic Minimum Spanning Tree (ODMST) problem outputs a spanning tree, denoted by $\mathcal{M}^t_{\text{ALG}}$, after obtaining $\sigma(t)$. This tree has to be a minimal spanning tree with respect to edge weights defined by $w^t(\cdot)$. The algorithm's cost in time step t is defined as the number of edges in which $\mathcal{M}^{t-1}_{\text{ALG}}$ and $\mathcal{M}^t_{\text{ALG}}$ differ, formally

$$C^t_{\text{ALG}} := \left| \left\{ e \in E \,|\, e \notin \mathcal{M}^{t-1}_{\text{ALG}} \wedge e \in \mathcal{M}^t_{\text{ALG}} \right\} \right| .$$

Consequently, $C_{\text{ALG}}(\sigma)$ is the cost of the algorithm on the whole sequence

$$C_{\text{ALG}}(\sigma) = \sum_{i=1}^{|\sigma|} C^i_{\text{ALG}} .$$

Although this does not change the complexity of the ODMST problem in any significant way, we allow the adversary to choose $\mathcal{M}^0_{\text{ALG}}$.

The optimal offline solution to the ODMST problem is defined by a sequence of spanning trees $\mathcal{M}^1_{\text{OPT}}, \ldots, \mathcal{M}^{|\sigma|}_{\text{OPT}}$, such that $\mathcal{M}^t_{\text{OPT}}$ is a minimal spanning tree with respect to edge weights defined by $w^t(\cdot)$. Given this restriction the sequence minimizes the value of $C_{\text{OPT}}(\sigma)$, with $C_{\text{OPT}}(\sigma) = \sum_{i=1}^{|\sigma|} C^i_{\text{OPT}}$, where

$$C^t_{\text{OPT}} := \left| \left\{ e \in E | e \notin \mathcal{M}^{t-1}_{\text{OPT}} \wedge e \in \mathcal{M}^t_{\text{OPT}} \right\} \right| .$$

As commonly found in literature, we will refer to the sequence $\mathcal{M}^1_{\text{OPT}}, \ldots, \mathcal{M}^{|\sigma|}_{\text{OPT}}$ as computed by an optimal algorithm OPT. This algorithms works in parallel to ALG, but has access to the complete sequence σ in advance and therefore can make optimal decisions in each time step.

Recall, that a deterministic algorithm ALG has a competitive ratio of $\mathcal{R}_{\text{ALG}}$ if for all input sequences σ we have $C_{\text{ALG}}(\sigma) \leq \mathcal{R}_{\text{ALG}} \cdot C_{\text{OPT}}(\sigma) + c$ or $\mathbf{E}[C_{\text{ALG}}(\sigma)] \leq \mathcal{R}_{\text{ALG}} \cdot C_{\text{OPT}}(\sigma) + c$ for a randomized algorithm. While the constant c does not depend on the input sequence, it can depend on the size of the input graph G.

Restricted ODMST problem Part of our results shown in this chapter will hold for the Restricted ODMST problem. In the case of the restricted version of this problem we allow the adversary to only increase the weight of the edges. Formally, the set of input sequences is limited to those fulfilling $\sigma(i) \in E \times \{+1\}$.

Stochastic adversaries We introduce two notions of stochastic adversaries, described by two stochastic processes. In the independent stochastic process defined in Section 6.6 the weight of an edge $e \in E$ is increased in each time step independently from other edges with some probability p_e. Then a transformation is applied to the constructed sequence, so that only one edge weight is increased per time step. This is necessary to match the definition of the ODMST problem.

The stochastic adversary is parameterizable and by an appropriate choice of the probabilities, it can be tailored to specific needs. Thereby, a wide variety of input sequences is modeled by the stochastic processes we describe here. We will say that a deterministic algorithm ALG solving the ODMST problem has a competitive ratio of $\mathcal{R}_{\text{ALG}}$ with a probability of $1 - \gamma$ if

$$\mathbf{Pr}_{\sigma}\left[C_{\text{ALG}}(\sigma) \leq \mathcal{R}_{\text{ALG}} \cdot C_{\text{OPT}}(\sigma) + c\right] \geq 1 - \gamma \ .$$

The described probability is calculated with respect to the probability distribution of input sequences, as implicitly defined by a stochastic process generating those sequences. As earlier, the constant c may not depend on the input sequence but only on the input graph G.

Notation The following notation will be used throughout the whole chapter. The set of alternative edges $\mathcal{A}(e,t)$ is defined for a graph $G = (V,E)$, a time step t, an algorithm ALG and an edge $e \in \mathcal{M}^t_{\text{ALG}}$. Removing e from $\mathcal{M}^t_{\text{ALG}}$ splits the tree into two parts. Consider the vertex sets V_1 and V_2 of both parts. Then the set of edges on the cut between V_1 and V_2 is denoted by

$$\mathcal{A}(e,t) = \{(u,v) \in E \mid u \in V_1 \wedge v \in V_2\} \ .$$

Consequently, the set of alternatives which have a certain weight is defined as

$$\mathcal{A}_w(e,t,w) = \{e' \in \mathcal{A}(e,t) \mid w^t(e') = w\} \ .$$

Furthermore we define $\mathcal{A}_w(\cdot)$ also for sets of edges, so that

$$\mathcal{A}_w(U, t, w) = \bigcup_{e \in U} \mathcal{A}_w(e, t, w) .$$

Suppose we extend $\mathcal{M}^t_{ALG}$ by adding an edge $e \in E$ and thus creating a cycle in $\mathcal{M}^t_{ALG}$. Then all edges on this cycle except for e are denoted by $\mathcal{K}(e, t)$. Analogously to the set $\mathcal{A}_w(\cdot)$, we define a set of all edges from $\mathcal{K}(e, t)$ with a certain weight

$$\mathcal{K}_w(e, t, w) = \{e' \in \mathcal{K}(e, t) \mid w^t(e') = w\} .$$

Note that $\mathcal{A}(e, t)$ includes e, whereas $\mathcal{K}(e, t)$ does not.

Results and Outline

In this chapter we will consider the ODMST problem in three different settings: for deterministic algorithms and randomized algorithms against a worst-case adversary and deterministic algorithms against a stochastic adversary. In Section 6.1 we show that any deterministic online algorithm has a competitive ratio of at least $\Omega(n^2)$, both for the ODMST and the Restricted ODMST problem. These results show, that on general graphs and without using randomization, online algorithms are inherently very bad against a worst-case adversary. To gain insight into the problem, in Section 6.2 we show a deterministic algorithm which achieves a competitive ratio of $n^2/2$.

As we will show, an improvement of the competitive ratio is possible for the Restricted ODMST problem if randomization can be used. As shown in Section 6.3, the randomized algorithm RANDMST is able to achieve an expected competitive ratio of $O(n \log n)$ on general graphs. Later we extend its analysis to the restricted case of planar input graphs and show that then it has an expected competitive ratio of only $O(\log n)$. In Section 6.1 we show a lower bound for randomized algorithms and the Restricted ODMST problem of $\Omega(\log n)$, so that the RANDMST algorithm on planar graphs is proven to be optimal up to a constant factor.

Another possibility to restrict the adversary and improve the results of algorithms solving the Restricted ODMST problem is to allow only a certain type of input sequence/input graph combination. Particularly, we restrict the adversary to generate only sequences, such that the number of edges with a certain weight in a time step is constant. We will call such input sequences weight-diversified and show in Section 6.5 that the RANDMST algorithm has constant competitive ratio on such input.

At last we show that weight-diversified sequences have practical relevance, in that a natural, highly parameterizable, stochastic adversary constructs such sequences with high probability. This implies, that the competitive ratio of RANDMST on sequences generated by this stochastic adversary is constant, as shown in Section 6.6.

Related work

Research on minimum spanning trees dates back to the well-known textbook algorithms by Kruskal [Kru56] and Prim [Pri57]. In the static setting improved solutions have been considered e.g in [CT76]. All this work assumes that the graph remains static and considers the classical runtime complexity of algorithms. Research in this area is still vivid, see e.g. recent results by Chazelle [Cha00] and Pettie [PR02].

Large effort has been put into constructing data structures which also allow minimum spanning trees to be computed efficiently when changes in the structure of the graph occur. These changes can either concern edge weights as assumed in our work (see e.g. [Fre83]) or might encompass adding and deleting vertices ([CH78, EGIN97, HK97]). Furthermore kinetic spanning trees have been considered in [AEGH98] to model the changes of edge lengths in a more predictable way. This allows planning of changes in the minimum spanning tree in advance, in comparison to a strictly online problem where the future cannot be foreseen.

An online problem related to ours has been presented in [RSS05], as similarly MSTs are considered in an online framework. The authors analyze the competitiveness of a minimum spanning tree algorithm which receives the weights of edges of a random graph as a sequence and has to decide immediately whether to include an edge into the MST or not. For an in-depth survey of different types of subproblems in the area of minimum spanning trees, for applications and results we refer the interested reader to [Epp96].

Similar problems have been also studied for shortest path trees, e.g. in [NST00, NST01, XCZ+04], where a shortest path tree is maintained under dynamics of the system. However, similarly to the case for the spanning tree maintenance, the solutions only consider the computational effort necessary for computing new shortest path trees.

Model Restrictions

Our model assumes that only one edge weight is changed per time step. This implies, that changes in the graph occur at a slow rate only. Such an assumption is typical also for other problems and models the scenarios in which demands may vary, but the pace of changes is restricted. Concrete examples are the k-server [MMS90] problem or the page migration and replication [BS89] problem.

The model of the ODMST problem assumes that the cost of changing an edge in the MST is unary, disregarding the distance between the removed and inserted edge. Assuming that the edges of the MST correspond to communication chains, this does not reflect the full energy cost, as the distance for the relay stations to travel depends on the distance between both edges. Therefore, we have to be aware that we are dealing with a simplification here.

At last, the randomized algorithm RANDMST works only for a restricted scenario, where the weights of edges can only grow. In this context, it is worth mentioning that the lower bounds presented in this chapter do not need to decrease edge weights. This gives some indication that the real hardness of the problem does not lie within decreasing edges, but can be also expressed by only increasing the weights of edges.

6.1. Lower bounds

In this section we show two lower bounds for the ODMST problem – one for deterministic algorithms and one for randomized ones.

For the deterministic case, we construct an input sequence for the ODMST problem which causes every online deterministic algorithm to have a large competitive ratio. We assume that the input sequence is given by an adversary who examines the moves of ALG. To construct a deterministic lower bound of $\mathcal{R}_{\text{ALG}}$ we have to be able to construct an input sequence σ for any algorithm ALG and any k such that $C_{\text{OPT}}(\sigma) \geq k$ and $C_{\text{ALG}}(\sigma) \geq \mathcal{R}_{\text{ALG}} \cdot C_{\text{OPT}}(\sigma)$. This is analogous to the formulation used in the Yao minmax principle [CLLR97, Yao77], simplified for the deterministic case here. Obviously, the following result holds also for algorithms solving the plain ODMST problem.

Theorem 6.1. *Let* ALG *be a deterministic algorithm solving the Restricted ODMST problem on graph G. Then for its competitive ratio* $\mathcal{R}_{\text{ALG}}$ *it holds* $\mathcal{R}_{\text{ALG}} \geq \Omega(n^2)$.

For the randomized case, we utilize the already mentioned Yao minmax principle. Thereby, we have to construct a probability distribution π over all input sequences and show that every deterministic offline algorithm knowing π has an expected competitive ratio of at least $\Omega(\log n)$, where the expectation is calculated over all input sequences distributed according to π.

Theorem 6.2. *Let* ALG *be a (possibly randomized) algorithm for the Restricted ODMST problem on graph G. Then for its expected competitive ratio* $\mathcal{R}_{\text{ALG}}$ *it holds* $\mathcal{R}_{\text{ALG}} \geq \Omega(\log n)$ *when the input sequence is constructed by an oblivious adversary.*

The construction of the lower bounds is, to some extent, similar. Therefore we will first define the common basis for both lower bounds.

Input graph. The input graph for the construction of the lower bounds is a complete graph $G = (V, E)$ with $|V|$. Partition V into two sets V_1 and V_2 with $|V_1| = |V_2|$. Call E_C the set of edges lying on the cut between V_1 and V_2. To each edge $e \in E_C$ we assign a weight $w^0(e) = n = |V|$, all other edges are assigned a weight of 1.

Obviously at least one edge from E_C must be used in every spanning tree and, since we consider minimum spanning trees, it will be the only one.

Input sequence. We construct an input sequence consisting of phases of length $|E_C|$. In each phase, the weight of all the edges from E_C is increased by one. Therefore, at the beginning of the p-th phase all edges from E_C have weight $n + p - 1$. As the edges in E_C have weight far larger then the weights of edges inside V_1 and V_2, each MST has to contain exactly one of the edges from E_C.

Generally, we restrict the order in which the weight of edges is increased during a phase only in one way. Let $e \in E_C$ be the edge whose weight has been increased in the last time step of phase p. Obviously, at the end of phase p it holds $e \in \mathcal{M}_{\text{ALG}}$ and $e \in \mathcal{M}_{\text{OPT}}$, as all edges from E_C have weight larger than e. We now require the weight of edge to be increased first in phase $p + 1$. This implies, that $e \notin \mathcal{M}_{\text{OPT}}$ after the increase and Opt has cost at least 1 during the phase. As Opt has access to the input sequence in advance, it can determine the edge whose weight is increased last in phase $p + 1$ at its beginning – and use that edge in $\mathcal{M}_{\text{OPT}}$ from the beginning on. Therefore, Opt's cost is exactly one in all phases but the first one. Thereby for any given k we can construct an input sequence σ by concatenating $k + 1$ phases, obtaining a σ such that $C_{\text{OPT}}(\sigma) \geq k$.

In the remaining part we will specify the order in which edges are increased in more detail. As the construction is different in order to obtain the deterministic and randomized lower bounds, we split our analysis from this point on.

6.1.1. Deterministic lower bound

As we are dealing with a deterministic algorithm Alg, the adversary can simulate its behavior on the input it provides, and therefore always knows $\mathcal{M}_{\text{ALG}}$. This allows her to increase the weight of the edge from E_C used in $\mathcal{M}_{\text{ALG}}$ in each time step, forcing the algorithm to perform $|E_C| - 1$ changes in $\mathcal{M}_{\text{ALG}}$ during a phase. This obviously incurs a cost of $|E_C| - 1$ during a phase. Combining with the fact, that Opt has unit cost per phase, Theorem 6.1 follows easily.

6.1.2. Randomized lower bound

For the randomized lower bound the order in which the edges' weights are increased is chosen randomly. In each phase, we uniformly at random choose one of the sequences

which increases the weight of all edges belonging to E_C. Obviously, this is a permutation of all edges from E_C, chosen uniformly at random.

With this random process defined, we have to give a bound on the number of edge changes that ALG has to perform in expectation in each phase. Fix some phase p. Let t be a time step within phase p, such that ALG performs a change in $\mathcal{M}_{\text{ALG}}$ in this time step. Denote by k the number of remaining edges with weight $n+p-1$ in E_C at time step t, whereas the remaining edges have weight $n+p$.

In time step t the algorithm has chosen a new edge e from E_C to be used in $\mathcal{M}_{\text{ALG}}$. Let $t+i$ be an upcoming time step within the same phase. Provided that the weight of edge e has not been increased since time step t, the probability that e's weight is increased in time step $t+i$ is exactly $1/(k-i+1)$. This is, since in each time step the edge whose weight is increased is chosen uniformly at random among those with weight $n+p-1$. Therefore, the unconditioned probability that the weight of e is increased in time step $t+i$ for any $i = 1,\ldots,k-1$ is equal to

$$\left(\frac{k-1}{k}\right)\cdot\left(\frac{k-2}{k-1}\right)\cdot\ldots\cdot\left(\frac{k-i}{k-i+1}\right)\cdot\frac{1}{k-i}=\frac{1}{k}\,.$$

We now define T_k to be the expected number of edge changes which ALG has to perform in the rest of the current phase, including the edge change performed in time step t. We can define recursively

$$T_k = 1+\sum_{i=0}^{k-1}\mathbf{Pr}[\text{ next edge change occurs in time step } t+i\,]\cdot T_i\,,$$

and by the earlier observation

$$T_k = 1+\frac{1}{k}\sum_{i=0}^{k-1}T_i.$$

Obviously $T_1 = 0$, since if there are no alternatives with weight n, increasing the last edge does not force the algorithm to any changes in $\mathcal{M}_{\text{ALG}}$. Adding $\sum_{i=1}^{k-1}T_i$ to both sides of the earlier equation we have

$$\sum_{i=1}^{k}T_i = 1+\sum_{i=1}^{k-1}\frac{k+1}{k}\cdot T_i\,.$$

By dividing both sides by $k+1$ we obtain

$$\frac{1}{k+1}\sum_{i=1}^{k}T_i = \frac{1}{k+1}+\sum_{i=0}^{k-1}\frac{1}{k}\cdot T_i\,.$$

We apply the definitions of T_{k+1} to the left side, and that of T_k to the right side and obtain

$$T_{k+1} = \frac{1}{k+1} + T_k = H_{k+1} = \Theta(\log k) .$$

Therefore, the expected number of changes in $\mathcal{M}_{\text{ALG}}$ during a whole phase is $T_{|E_C|-1} = \Theta(\log |E_C| - 1)$. As $|E_C| = n^2/2$ we can easily follow Theorem 6.2.

Planar graphs. Note, that one can apply the construction presented here even if the underlying graph G is restricted to be planar. In that case one can construct a graph G such that the number of edges in E_C is $\Theta(n)$. Therefore the following corollary holds.

Corollary 6.3. *Let* Alg *be a (possibly randomized) algorithm for the Restricted ODMST problem on planar graph G. Then for its expected competitive ratio* $\mathcal{R}_{\text{ALG}}$ *it holds* $\mathcal{R}_{\text{ALG}} \geq \Omega(\log n)$ *when the input sequence is constructed by an oblivious adversary.*

6.2. Algorithm MSTMark

In this section we present the deterministic algorithm MSTMark which achieves an optimal, up to a constant factor, competitive ratio for the ODMST problem, as summarized by the following theorem.

Theorem 6.4. *For the competitive ratio of algorithm* MSTMark *it holds* $\mathcal{R}_{\text{MSTMark}} \leq n^2/2$.

Notation. The MSTMark algorithm (Algorithm 1) works on a graph $G = (V, E)$ computing a minimum spanning tree $\mathcal{M}^t_{\text{ALG}}$ in each time step t. Where clear from the context we will write $\mathcal{M}_{\text{ALG}}$ instead of $\mathcal{M}^t_{\text{ALG}}$ omitting the current time step number. Analogously we will omit the superscript in the notation $\mathcal{M}_{\text{OPT}}$ wherever convenient. We say that an algorithm substitutes edge e with f in time step t if we have $\mathcal{M}^{t+1}_{\text{ALG}} = (\mathcal{M}^t_{\text{ALG}} \setminus \{e\}) \cup \{f\}$.

MSTMark algorithm. The algorithm has to respond to two different kinds of events – increases and decreases of weights of edges in G. We will be following a greedy approach, which changes only one edge of the MST when ultimatively necessary. If the weight of an edge $e \in \mathcal{M}^{t-1}_{\text{ALG}}$ is increased in time step t, MSTMark tries to find a suitable alternative $f \in \mathcal{A}_w(e, t-1, w^{t-1}(e))$. If a not marked edge f can be found, MSTMark replaces e with f in $\mathcal{M}^t_{\text{ALG}}$. By the construction of the set $\mathcal{A}_w(\cdot)$ any such edge causes $\mathcal{M}_{\text{ALG}}$ to remain a minimum spanning tree. If an appropriate edge cannot be found, MSTMark sets $\mathcal{M}^t_{\text{ALG}} = \mathcal{M}^{t-1}_{\text{ALG}}$.

Algorithm 1 MSTMARK(time step t)

if weight of edge $e \in \mathcal{M}_{\text{ALG}}^{t-1}$ increased **and** $\mathcal{A}_{\text{w}}(e, t-1, w^{t-1}(e)) \neq \emptyset$ **then**
  $\mathcal{A}_{NM} \leftarrow \{f \in \mathcal{A}_{\text{w}}(e, t-1, w^{t-1}(e)) | f$ isn't marked with ABSENCE$\}$
  if $\mathcal{A}_{NM} \neq \emptyset$ **then**
    remove e from $\mathcal{M}_{\text{ALG}}^{t}$ and substitute it with any $f \in \mathcal{A}_{NM}$
    mark e with ABSENCE
  else
    remove e from $\mathcal{M}_{\text{ALG}}^{t}$ and substitute it with $f \in \mathcal{A}_{\text{w}}(e, t-1, w^{t-1}(e))$
    remove all marks
    mark e with ABSENCE
  end if
end if
if weight of edge $e \notin \mathcal{M}_{\text{ALG}}^{t-1}$ decreased **and** $\mathcal{K}_{\text{w}}(e, t-1, w^{t-1}(e)) \neq \emptyset$ **then**
  $\mathcal{K}_{NM} \leftarrow \{f \in \mathcal{K}_{\text{w}}(e, t-1, w^{t-1}(e)) | f$ isn't marked with PRESENCE$\}$
  if $\mathcal{K}_{NM} \neq \emptyset$ **then**
    remove e from $\mathcal{M}_{\text{ALG}}^{t}$ and substitute it with any $f \in \mathcal{K}_{NM}$
    mark f with PRESENCE
  else
    remove e from $\mathcal{M}_{\text{ALG}}^{t}$ and substitute it with $f \in \mathcal{A}_{\text{w}}(e, t-1, w^{t-1}(e))$
    remove all marks
    mark f with PRESENCE
  end if
end if
if e is marked with two different flags **then**
  remove all marks
  mark e with flag it obtained earlier this time step
end if

If the weight of an edge $e \notin \mathcal{M}_{\text{ALG}}^{t-1}$ is decreased in time step t, MSTMARK checks whether there is a not marked edge $f \in \mathcal{K}(e, t-1)$ with a higher weight than $w^t(e)$. If yes, it substitutes f with e within $\mathcal{M}_{\text{ALG}}$. If no, MSTMARK sets $\mathcal{M}_{\text{ALG}}^{t} = \mathcal{M}_{\text{ALG}}^{t-1}$.

In all other cases MSTMARK does not perform any changes in its minimum spanning tree.

The greedy approach changing only one edge of the MST on updates of edge weight has been already successfully applied in algorithms for updating minimum spanning trees, e.g. in [Fre83], thus we won't argue its correctness.

Marking with flags. In addition to the described behavior, MSTMark marks edges of G with two kinds of flags: presence and absence. The idea is that a flag is put on an edge e, where MSTMark is sure that, respectively, $e \in \mathcal{M}_{\text{OPT}}$ or $e \notin \mathcal{M}_{\text{OPT}}$. This information is only guaranteed for the very time step when the mark has been set – for future time steps it may not hold anymore.

The presence flag is set on edge e if and only if the decrease of e's weight has caused the MST in the graph to decrease its weight. Similarly, the absence flag is only set on edge e, when its increase caused the MST to increase its weight. As it is easy for the MSTMark algorithm to determine these situations, we will only have to show that at this specific time of point we have respectively $e \in \mathcal{M}_{\text{OPT}}$ or $e \notin \mathcal{M}_{\text{OPT}}$.

Outline of analysis. For the analysis of the competitive ratio of MSTMark we introduce the notion of epochs. The presence and absence flags are the key to this analysis. An epoch starts when all flags are removed from the graph and lasts until the next removal of all flags.

The analysis of the competitive ratio of MSTMark proceeds with the following steps. First, we formally prove the role of the flags in Lemma 6.5. Then, Corollary 6.10 shows that Opt has at least unit cost in each epoch. Finally Lemma 6.11 establishes the fact that an epoch can last at most $O(n^2)$ time steps. From these lemmas we will derive Theorem 6.4.

Lemma 6.5. *If an edge e has flag* presence *set in time step t, then there exists a time step $z \leq t$ in the current epoch, such that $e \in \mathcal{M}^z_{\text{OPT}}$. Analogously, if it has the flag* absence *set, then there exists $z \leq t$ in the current epoch, s.t. $e \notin \mathcal{M}^z_{\text{OPT}}$.*

Proof. Assume that e has flag presence set. Then, there was some time step z in which the weight of edge e decreased and this caused MSTMark to include e into $\mathcal{M}^z_{\text{ALG}}$. This means that the weight of any MST decreased between time step $z-1$ and time step z. Opt either included $e \in \mathcal{M}^{z-1}_{\text{OPT}}$ – then the weight of $\mathcal{M}_{\text{OPT}}$ decreased automatically – or it had to include $e \in \mathcal{M}^z_{\text{OPT}}$ by removing another edge from $\mathcal{M}^{z-1}_{\text{OPT}}$. In both cases, edge e was eventually included in $\mathcal{M}^z_{\text{OPT}}$.

Now assume that e has flag absence set. Then, there has been a time step z in which the weight of edge e was increased. If e had not been included in $\mathcal{M}_{\text{OPT}}$ at time step $z-1$ then there is nothing to show. Since MSTMark found a suitable alternative edge if absence was set at e, the weight of $\mathcal{M}^z_{\text{ALG}}$ was equal to that of $\mathcal{M}^{z-1}_{\text{ALG}}$. Then, if e was included in $\mathcal{M}^{z-1}_{\text{OPT}}$, it could not stay the same in time step z – the weight of $\mathcal{M}^z_{\text{OPT}}$ would increase and become greater than $\mathcal{M}^z_{\text{ALG}}$. ■

Now we turn our attention to lower bounding OPT's cost in an epoch. If an epoch ends because of an edge marked with two different kinds of flags (line 24), then we have an immediate proof that at least one edge changed in $\mathcal{M}_{\text{OPT}}$ during the current epoch.

In Lemma 6.7 we will establish the correctness of the fact that at least one edge change in $\mathcal{M}_{\text{OPT}}$ has occurred within one epoch if it ends with line 19. Beforehand, we need to introduce a short technical lemma.

Lemma 6.6. *Consider an edge $e \notin \mathcal{M}^t_{\text{ALG}}$. Then for all $f \in \mathcal{K}(e,t)$ it holds $w^t(f) \leq w^t(e)$.*

Proof. Assume there exists an edge $f \in \mathcal{K}(e,t)$ with $w^t(f) > w^t(e)$. Then we could substitute f with e and preserve the spanning property of $\mathcal{M}^t_{\text{ALG}}$. This would decrease the weight of $\mathcal{M}^t_{\text{ALG}}$ which contradicts its minimum property. ■

Lemma 6.7. *Assume that $\delta(t) = (e,-1)$, such that $e \notin \mathcal{M}^{t-1}_{\text{ALG}}$ and that this causes the weight of minimal spanning trees to decrease. Furthermore assume that all edges within $\mathcal{K}_{\text{w}}(e, t-1, w^{t-1}(e))$ have flag* PRESENCE *set. Then there exists a time step $i \leq t$ in the current epoch such that $C^i_{\text{OPT}} > 0$.*

Proof. According to the assumptions of the lemma the weight of edge e is decreased in time step t. Since the weight of all MSTs has decreased, $e \in \mathcal{M}^t_{\text{OPT}}$ after the decrease. Removing e from $\mathcal{M}^t_{\text{OPT}}$ causes the spanning tree of OPT to be split into two connected components. Call them $M_1 = (V_1, E_1)$ and $M_2 = (V_2, E_2)$. Figure 6.1 illustrates this division.

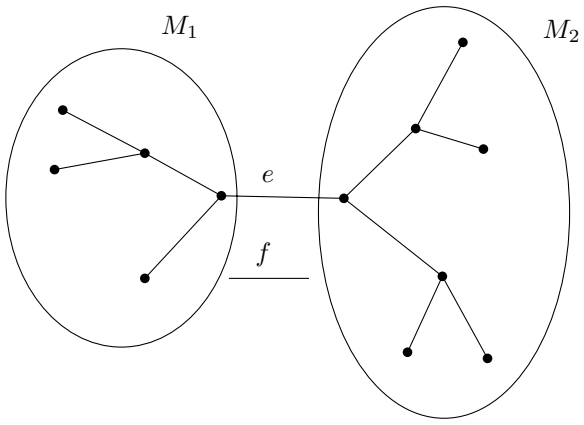


Figure 6.1.: OPT's tree divided into two parts by edge e. Edge f is on the cycle created in $\mathcal{M}_{\text{ALG}}$ by e and must have weight equal to that of e.

There are two cases: either $e \in \mathcal{M}_{\text{OPT}}^{t-1}$ (before the decrease), or some other edge from the cut between V_1 and V_2 was in $\mathcal{M}_{\text{OPT}}^{t-1}$. If the second case occurs, then Opt has to remove some edge from $\mathcal{M}_{\text{OPT}}^{t-1}$ and replace it with e, thus having cost 1 in the current time step.

Assume that $e \in \mathcal{M}_{\text{OPT}}^{t-1}$ (before the decrease). We now turn our attention to $\mathcal{M}_{\text{ALG}}$. Obviously after the decrease $e \in \mathcal{M}_{\text{ALG}}^{t}$, but according to the assumptions of the lemma $e \notin \mathcal{M}_{\text{ALG}}^{t-1}$. As e connects V_1 and V_2, at least one of the edges from the cycle $\mathcal{K}(e, t-1)$ lies on the cut between V_1 and V_2. Call it f. Since this edge is in $\mathcal{K}(e, t-1)$, it holds $w^t(f) = w^{t-1}(f) \leq w^{t-1}(e)$ according to Lemma 6.6. Now since this edge lies on the cut between V_1 and V_2 and we have assumed that $e \in \mathcal{M}_{\text{OPT}}^{t-1}$, f cannot have weight smaller than $w^{t-1}(e)$, since then it would have been used by Opt in round $t-1$ in $\mathcal{M}_{\text{OPT}}^{t-1}$. Thus, $w^t(f) = w^{t-1}(e)$ and $f \in \mathcal{K}_{\text{w}}(e, t-1, w^{t-1}(e))$ by the definition of $\mathcal{K}_{\text{w}}(\cdot)$.

According to the assumption of the lemma, all edges in $\mathcal{K}_{\text{w}}(e, t-1, w^{t-1}(e))$, in particular f, have the flag presence set. So at some time step in the current epoch $f \in \mathcal{M}_{\text{OPT}}$ but now $f \notin \mathcal{M}_{\text{OPT}}$. This leads to the conclusion that Opt has paid at least once during the current epoch. ■

Analogically to the reasoning above, we can state Lemma 6.9 showing that at least one edge change has occurred in $\mathcal{M}_{\text{OPT}}$ during an epoch, if it ends in line 8 of MSTMark. Prior to that we introduce another technical lemma.

Lemma 6.8. *Consider an edge $e \in \mathcal{M}_{\text{ALG}}^{t}$ and its set of alternative edges $\mathcal{A}(e, t)$. Then there exists an edge $f \in \mathcal{A}(e, t)$ with $f \in \mathcal{M}_{\text{OPT}}^{t}$ and $w^t(f) = w^t(e)$.*

Proof. Obviously there must be at least one $f \in \mathcal{A}(e, t)$ with $f \in \mathcal{M}_{\text{OPT}}^{t}$ (recall that $e \in \mathcal{A}(e, t)$). We have to show that it has the same weight as e. If $e = f$ the fact follows trivially.

Assume $e \neq f$. If $w^t(f)$ was smaller than $w^t(e)$, then we could substitute e with f in $\mathcal{M}_{\text{ALG}}^{t}$ and thus we would obtain a spanning tree with smaller weight – a contradiction since $\mathcal{M}_{\text{ALG}}^{t}$ is a minimum spanning tree.

Now assume $w^t(f)$ is greater than $w^t(e)$. Then increase the weight of all edges in $\mathcal{A}_{\text{w}}(e, t, w^t(e))$. This increases the weight of $\mathcal{M}_{\text{ALG}}$, since the weight of e and all of its alternatives have been increased. As none of the edges in $\mathcal{A}_{\text{w}}(e, t, w^t(e))$ belongs to $\mathcal{M}_{\text{OPT}}^{t}$, increasing their weight won't change the weight of $\mathcal{M}_{\text{OPT}}^{t}$. This is a contradiction, since both $\mathcal{M}_{\text{OPT}}^{t}$ and $\mathcal{M}_{\text{ALG}}^{t}$ are minimum spanning trees and must have the same weight all the time. ■

Lemma 6.9. *Fix an edge $e \in \mathcal{M}^t_{\text{ALG}}$. Assume all $f \in \mathcal{A}_w(e, t, w^t(e))$ have flag* ABSENCE *set. Then* OPT *has incured at least unit cost during the current epoch.*

Proof. According to the presented Lemma 6.8, OPT must have at least one edge in $\mathcal{A}_w(e, t, w^t(e))$. Let it be f. Since f has flag ABSENCE set, OPT has changed at least one edge in $\mathcal{M}_{\text{OPT}}$ within the current epoch. ■

Combining both Lemma 6.7 and Lemma 6.9 the corollary follows.

Corollary 6.10. *During an epoch* OPT *incurs at least unit cost.*

In every time step in which MSTMARK pays for changing its edge it also marks one edge with PRESENCE or ABSENCE. Lemma 6.11 establishes the fact that MSTMARK doesn't mark any edges twice and allows us to conclude that MSTMARK pays only $|E| \leq n^2/2$ times within one epoch. Then Theorem 6.4 follows easily.

Lemma 6.11. *No edge is marked more than once within one epoch.*

Proof. By contradiction, assume that an edge e is marked twice with flag ABSENCE. This would mean, that the weight of this edge was increased twice and every time it belonged to $\mathcal{M}_{\text{ALG}}$. Directly after it was marked for the first time it was removed from $\mathcal{M}_{\text{ALG}}$. Consider the first possibility which would cause e to be included in $\mathcal{M}_{\text{ALG}}$ again, namely that the weight of e was decreased and this caused the weight of the minimum spanning tree to decrease. But then e would be marked with PRESENCE and this causes all marks to be removed (see MSTMARK step 24). The other possibility is that the weight of some edge f has been increased and then e was chosen as an alternative for f. But if MSTMARK chooses an edge with ABSENCE as an alternative it also removes all marks (see MSTMARK step 19).

A similar reasoning can be applied to flags of type PRESENCE. Assume that an edge e has been marked twice with flag PRESENCE. This means that the weight of e has been decreased twice and both times it wasn't included in $\mathcal{M}_{\text{ALG}}$. Thus, there was some time step between these two in which e was removed from $\mathcal{M}_{\text{ALG}}$. The first possibility is that the weight of e has been increased. But then e would be marked with ABSENCE and this causes all marks to be removed (see MSTMARK step 24). The other possibility is that the weight of some other edge f has been decreased and e was on the cycle in $\mathcal{M}_{\text{ALG}}$ created

by f. But if MSTMARK chooses an edge from a cycle with PRESENCE set, it also removes all marks (see MSTMARK step 8).

It is clear due to line 24 of MSTMARK that no edge can be first marked with one flag type and later with the second type within one epoch. ■

6.3. Randomized algorithm RANDMST

The randomized algorithm RANDMST presented in this section achieves an expected competitive ratio of $O(n \log n)$ on the Restricted ODMST problem for general input graphs. It is a cut down version of the MSTMARK algorithm, with the handling of flags removed. In the considered scenario edge weights can only grow and we will see that flags are not necessary any more.

Algorithm 2 RANDMST

1: **if** weight of edge $e \in \mathcal{M}_{\text{ALG}}^{t-1}$ increased in time step t **and** $\mathcal{A}_{\text{w}}(e, t-1, w^{t-1}(e)) \neq \emptyset$ **then**
2: $f \leftarrow$ choose uniformly at random an edge out of $\mathcal{A}_{\text{w}}(e, t-1, w^{t-1}(e))$
3: remove e from $\mathcal{M}_{\text{ALG}}^{t}$ and substitute it with f
4: **end if**

Consider a time step t in which the weight of an edge $e \in \mathcal{M}_{\text{ALG}}$ is increased. If there exist alternative edges for e with weight $w^{t-1}(e)$, then RANDMST selects one of these edges uniformly at random and uses it instead of e in $\mathcal{M}_{\text{ALG}}^{t}$. In other cases RANDMST ignores the edge weight increase.

Outline of the analysis. The idea of the analysis is to consider the behavior of RANDMST in *layers* separately. Intuitively, a layer w consists only of edges which have weight w. In every layer we will divide the graph G into *fixed components* and *edge sets*, connecting those fixed component. We will show that creating a fixed component corresponds to some cost incurred by OPT, and that therefore the cost of OPT is at least proportional to the number of fixed components created in all layers. We will also be able to bound the expected cost of RANDMST to $O(n \log n)$ times the number of fixed components created. From this we will be able to conclude that the expected competitive ratio of RANDMST is at most $O(n \log n)$. Note that the layers and fixed components are structures used purely for the analysis, RANDMST's execution does not depend on them.

Theorem 6.12. *For the expected competitive ratio of algorithm* RANDMST *it holds* $\mathcal{R}_{\text{RandMST}} \leq O(n \log n)$.

Fixed components. As already mentioned, we consider a separate layer of fixed components for each weight w. Let X be a subgraph of G. Then $V(X)$ denotes the set of vertices of X and $E(X)$ its edges. A fixed component is a subgraph of G. Prior to the first time step there is exactly one fixed component in every layer containing the whole graph G. For a fixed component F the part of $\mathcal{M}_{\text{ALG}}$ contained in F is the subgraph with vertex set $V(F)$ and edge set $E(F) \cup E(\mathcal{M}_{\text{ALG}})$. A fixed component F in layer w splits if the weight of an edge $e \in E(F)$ with $w(e) = w$ is increased, $e \in \mathcal{M}_{\text{ALG}}$, and the size of minimum spanning trees does not increase (i.e. RANDMST must perform a change in $\mathcal{M}_{\text{ALG}}$). The edge e divides $\mathcal{M}_{\text{ALG}}$ into two parts – call the sets of vertices in each of these parts V_1 and V_2.

The fixed component F splits into two fixed components F_1 and F_2, such that $V(F_1) = V_1 \cup V(F)$ and $V(F_2) = V_2 \cup V(F)$. The edge sets of both components consist of the edges induced by their vertex sets. We say that a fixed component splits on edge e if the split occurs due to an increase of weight of e. Note, that fixed components form a partition of the vertex set of G, and thus there are at most n fixed components in any layer. Note also, that it is necessary for the splitting technique to work properly that the part of $\mathcal{M}_{\text{ALG}}$ contained in a fixed component prior to a split is connected. This property will be established in Lemma 6.15.

Besides fixed components, each layer also contains edge sets. Before time step 1 there are no edge sets in any layer. If a fixed component F splits into F_1 and F_2 on an edge e with weight w, an edge set between F_1 and F_2 is created, denoted by $E_S(F_1, F_2)$. It consists of all edges between vertices of F_1 and F_2 having weight w. If a fixed component F splits into F_1 and F_2, edge sets connected to F also split. Consider such an edge set $E_S(F, F')$. Then every edge $e \in E_S(F, F')$ is put either into $E_S(F_1, F')$ or $E_S(F_2, F')$ depending on whether this edge connects F' to F_1 or F_2. Note, that since there are at most n fixed components in a layer, the number of edge sets created during one split is upper bounded by $2n$.

Edges already contained in an edge set remain in an edge set after their weight has been increased.

Pseudo-splits. We allow fixed components also to pseudo-split in another situation. Consider a layer w and a time step t. Let e be an edge contained in a fixed component F, such that $w^{t-1}(e) > w$. The component F pseudo-splits, if the weight of edge e is increased in time step t and the reorganization of $\mathcal{M}_{\text{ALG}}$ in time step t causes the fixed component F to contain two disconnected parts of the tree $\mathcal{M}_{\text{ALG}}$. The pseudo-split is carried out exactly as in the case of a normal split, edge-sets are also divided in the same way as earlier.

Nevertheless, there is some crucial difference between a split and a pseudo-split. Although the pseudo-split can also create new edge sets, the new edge sets have no

practical relevance, as they do not contain any edges with weight w. This is formally established by the following two lemmas.

Lemma 6.13. *Let the fixed component F be divided by a pseudo-split into F_1 and F_2. Then the edge set $E_S(F_1, F_2)$ contains no edges.*

Proof. Assume that the fixed component F pseudo-splits into F_1 and F_2 on edge e in time step t. By definition, for edge e we have $w^{t-1}(e) > w$. Assume that after the pseudo-split, i.e. after removing e from $\mathcal{M}_{\text{ALG}}$, there exists an edge f between F_1 and F_2 which has weight w. Then we could have substituted e with f prior to the pseudo-split, decreasing the weight of $\mathcal{M}_{\text{ALG}}$. This is a contradiction. ■

Lemma 6.14. *Let X be an edge set divided into X_1 and X_2 by a pseudo-split of a fixed component F. Assume that the part of $\mathcal{M}_{\text{ALG}}$ contained in F is connected prior to the pseudo-split. Then at most one of X_1 and X_2 contains edges with weight w.*

Proof. Let F and F' be the fixed components connected originally by X. Let F pseudo-split into F_1 and F_2 on edge e.

As F was connected and therefore an edge from $E_S(F_1, F_2)$ with weight at least $w+1$ is used in $\mathcal{M}_{\text{ALG}}$. For the same reason, at most one of the edges from $E_S(F_1, F_2)$ is used in $\mathcal{M}_{\text{ALG}}$.

Let X by split into X_1 and X_2 and assume that both of them contain edges with weight w. As the part of $\mathcal{M}_{\text{ALG}}$ contained in F was connected prior to the pseudo-split, only one of the edges from X with weight w could have been used in $\mathcal{M}_{\text{ALG}}$. Without loss of generality assume that this edge is contained in X_1. Then, in time step $t-1$ we could have substituted edge e with an edge with weight w from X_2, obtaining a connected spanning tree with weight smaller than $\mathcal{M}_{\text{ALG}}$. This yields a contradiction. ■

Interaction between layers. We will now examine the interactions between distinct layers, the fixed components, and edge sets.

We want to show that these interactions follow certain rules and that a certain property (as expressed by Corollary 6.16) is always fulfilled in a layer.

Lemma 6.15. *Let e be an edge between two fixed components from layer w. Then $w^t(e) \geq w$. Furthermore, the part of $\mathcal{M}_{\text{ALG}}$ contained in a fixed component F is connected.*

Proof. We will prove this property by induction on time step number in each layer. Consider the induction basis in time step 1. By definition, there is one fixed component in each layer and therefore the statement of the lemma is fulfilled.

For the inductive step we assume that the lemma holds time step t and those earlier. We have now to show that it holds for time step $t+1$. In time step $t+1$ the adversary can perform the following actions: it increases the weight of some edge with weight w, it increases the weight of some edge with weight smaller than w or it increases the weight of some edge with weight larger than w. We will go through all three cases.

Assume that the adversary increases the weight of some edge e in layer w. The only change in the structure of fixed components occurs when the increase of weight of e causes a fixed component F to split. By inductive assumption F contained exactly one connected part of $\mathcal{M}_{\text{ALG}}$. The splitting procedure assures that both F_1 and F_2 created by splitting F contain exactly one connected part of $\mathcal{M}_{\text{ALG}}$. We only have to show, that there is no edge of weight smaller than e between F_1 and F_2. For the sake of contradiction assume that there is an edge with weight smaller than w between F_1 and F_2 – call it f. Then it is either used in $\mathcal{M}_{\text{ALG}}$ or is not. If it is used in $\mathcal{M}_{\text{ALG}}$, then by our previous reasoning in both F_1 and F_2 the parts of $\mathcal{M}_{\text{ALG}}$ are connected. But having two edges e and f between F_1 and F_2 would cause a loop. On the other hand, if f is not used in $\mathcal{M}_{\text{ALG}}$, then using f instead of e would cause the weight of $\mathcal{M}_{\text{ALG}}$ to decrease – this is a contradiction since $\mathcal{M}_{\text{ALG}}$ is a minimum spanning tree.

Now we have to turn to the case when the adversary increases the weight of some edge e in a layer smaller than w. By the inductive assumption no edge of this weight connects two fixed components from layer w, thus e lays inside of a fixed component F from layer w. This operation could potentially cause the part of $\mathcal{M}_{\text{ALG}}$ contained in F to become disconnected. For this to happen, an alternative edge f for e must be found which does not lie in F and has weight smaller than w. Since f does not lie in F, it connects F to another fixed component and thus cannot have weight smaller than w.

Let us get our focus to the third case – the weight of some edge e with weight larger than w is increased. If the edge lies between any two fixed components in layer w then the increase cannot affect any fixed component in layer w. Thus, assume that e is contained within a fixed component F in layer w. Assume also that this increase would disconnect the part of $\mathcal{M}_{\text{ALG}}$ contained in F into two trees M_1 and M_2. Therefore, all prerequisites for pseudo-splitting F are fulfilled and the new fixed components contain connected parts of $\mathcal{M}_{\text{ALG}}$. ■

The following corollary states the most important fact for the analysis of the competitive ratio of RANDMST.

Corollary 6.16. *The* RANDMST *algorithm uses at most one edge of one edge set in* $\mathcal{M}_{\text{ALG}}$.

Splits and OPT's cost. We want to establish a bound between the number of operations OPT performs on some layer w and the number of fixed component splits which have occurred on this layer. For this purpose let $\#E_w(G)$ denote the number of edges with weight w in the input graph at the beginning of the execution.

Lemma 6.17. *Let* s_w *be the number of fixed component splits in layer* w *during the whole execution of an input sequence. Then* OPT *has a cost of at least* $s_w - \#E_w(G)$ *in layer* w.

Proof. Let e be an edge on which a split in level w occurs in time step t. Obviously, at the moment when the split occurred, $e \in \mathcal{M}_{\text{ALG}}$. Let us consider when e has become included in $\mathcal{M}_{\text{ALG}}$. The first possibility is it was included in the initial MST, i.e. $\mathcal{M}^0_{\text{ALG}}$. The second possibility is that it has been included in $\mathcal{M}_{\text{ALG}}$ when the weight of e was smaller than w. Then let z be the time step when the weight of e has been increased to w. In time step z the weight of $\mathcal{M}_{\text{ALG}}$ increases, therefore it holds $e \in \mathcal{M}_{\text{OPT}}z - 1$. As with the increase of e's weight in step t the weight of the MST does not increase, OPT has to remove e from $\mathcal{M}_{\text{OPT}}$ at t at latest. We assign this movement by OPT to the split.

By this assignment, we can assign a unit of OPT's cost to all splits but for at most $\#E_w(G)$. Therefore OPT's cost on the layer w is at least $s_w - \#E_w(G)$. ■

By the last lemma, nearly every fixed component split (except for n splits for the whole execution) can be mapped to a time step which causes a cost of 1 to OPT. We will call a mapping of all of RANDMST's costs to fixed component splits a cost assignment scheme. The following corollary follows easily from Lemma 6.17.

Corollary 6.18. *Let there exist a scheme assigning all* RANDMST*'s cost to fixed component splits, such that each split is assigned in expectation at most* T *cost. Then* RANDMST *has an expected competitive ratio of* T.

Therefore, if we can assign RANDMST's costs to fixed component splits so that each split receives at most $O(n \log n)$ cost in expectation, then we can easily conclude that the expected competitive ratio of RANDMST is $O(n \log n)$.

Cost assignment scheme. Every time a split of a fixed component F into F_1 and F_2 occurs, we assign all created edge sets to this split. This also includes edge sets which

are divided in two by the split. This means that an edge set, which has previously been assigned to some split s be assigned to the split of F now. This operation can only decrease the number of edge sets assigned to any fixed component s. Since a fixed component split can create at most $2n$ edge sets, at most $2n$ edge sets are assigned to a split.

There are no edge sets assigned to pseudo-splits. Let X be an edge set divided by a pseudo-split into X_1 and X_2. By Lemma 6.14 only one of X_1 and X_2 contains edges with weight w. Let this be X_1. Then X_1 remains assigned to the split it was assigned to. The edge set X_2 becomes unassigned as it contains no edges with weight w and therefore no further cost will be assigned to it.

We still have to bound the cost of RANDMST on one edge set, i.e. bound the number of edge increases in an edge set which causes RANDMST to change an edge in $\mathcal{M}_{\text{ALG}}$.

Consider the way RANDMST chooses a new edge as an alternative for an edge e used before in layer w. This new edge is chosen from the whole alternative set uniformly at random. This alternative set is at least as large as the number of edges with weight w in the current edge set. What is important, is that each of the edges in the current edge set is chosen with the same probability. Thus, even if the adversary knows the code of RANDMST and the probability distribution used by it, it can have no knowledge which particular edge is used within an edge set in $\mathcal{M}_{\text{ALG}}$. On the other hand, by Corollary 6.16 we know that at most one edge out of an edge set is used in $\mathcal{M}_{\text{ALG}}$.

Let p_{E_S} describe the probability that an edge out of the edge set E_S is used in $\mathcal{M}_{\text{ALG}}$. Let $\#E_S^w$ describe the number of edges with weight w in E_S. Assume that we are now increasing the weight of an edge in edge set E_S. Then, the probability of increasing the weight of an edge which is in $\mathcal{M}_{\text{ALG}}$ is exactly $p_{E_S} \cdot 1/\#E_S^w$. We can upper bound $p_{E_S} \leq 1$. Furthermore, we know that the probability of increasing the weight of an edge in $\mathcal{M}_{\text{ALG}}$ is equal to the expected cost of RANDMST, since RANDMST's cost is either 0 or 1.

To bound the expected cost of RANDMST on an edge set E_S situated in layer w, we only look at requests in the input sequence which increase the weight of an edge set. Each of these requests decreases the number of edges with weight w in E_S by one. What is important and follows from previous considerations, is that the number of edges with weight w in an edge set in layer w never increases after it has been created. So, the expected cost of RANDMST on E_S is then at most $\frac{1}{x} + \frac{1}{x-1} + \cdots + 1$, where x denotes the number of edges with weight w at the moment of the creation of E_S. This value is equal to $\Theta(\log n)$, since we can upper bound $x \leq n^2$.

This cost assignment scheme assures that every change of edges in $\mathcal{M}_{\text{ALG}}$ producing cost is assigned to one edge set and, on the other hand, this edge set is assigned to a fixed component split. From the fact, that each fixed component split is assigned at most $O(n)$ edge sets and that each of these edge sets receives an expected cost of $O(\log n)$ together

with Corollary 6.18 we can easily conclude Theorem 6.12.

6.4. Planar Graphs

The described algorithm RANDMST works with any graph G. In this section we want to restrict the input to planar graphs only. We show that RANDMST performs optimally, up to a constant factor, for these input graphs. For that purpose, we will show that in a planar graph the number of edge sets created is significantly lower than in general graphs. Therefore, we can assign all edge sets to fixed component splits so that each fixed component split is assigned only a constant number of edge sets.

Theorem 6.19. *For the expected competitive ratio of algorithm* RANDMST *on planar graphs it holds* $\mathcal{R}_{\text{RandMST}} \leq O(\log n)$.

First we want to show a property of planar graphs crucial for the analysis. We denote by *fixed component graph* a graph constructed out of a particular configuration of fixed components and edge sets in a layer. The fixed components are vertices in the fixed component graph and the edge sets are edges. The following theorem can be found in [MT01].

Theorem 6.20 (Kuratowski)**.** *A graph is planar if and only if it does not contain* K_5 *or* $K_{3,3}$ *as a minor.*

Lemma 6.21. *Let G be the graph for the ODMST problem. Denote by F a fixed component graph constructed on top of G. Then, if G is planar then F is planar too.*

Proof. A graph H is a minor of G if G can be transformed to H by contracting edges, deleting edges and deleting isolated vertices. The contraction operation, applied to an edge (u, v), removes the edge and merges the vertices u and v into one new vertex.

Denote by F a fixed component graph and G the underlying graph for the ODMST problem. We will prove the lemma by showing that if F is non-planar then G is non-planar too. So, assume F is not planar. Then F contains K_5 or $K_{3,3}$ as a minor. We can assume without loss of generality that K_5 is F's minor. Apply all edge and vertex deletions to F which are necessary to convert it to K_5 and call the resulting graph F'.

In order to obtain K_5 from F', all vertices of F' must be contracted to 5 vertices. Call V_i (for $i = 1, \ldots, 5$) the set of vertices of F' which are contracted to the i-th vertex of K_5. Any V_i (for $i = 1, \ldots, 5$) must be connected to all four V_j (for $j \neq i$), so there are at most 4 vertices in V_i which are connected via an edge to another V_j. All these vertices must be connected by edges within V_i so that the contraction can be performed.

Now consider each of the vertices of V_i as a fixed component composed of vertices from G. Among these vertices are those which are connected to V_j's. Since the subgraph of G contained in a fixed component is connected, and the fixed components within V_i are connected we can contract these vertices to one.

Repeating this procedure for every V_i contracts G to K_5. By Theorem 6.20 introduced earlier this implies that G is not planar. ■

The previous lemma allows to improve the analysis of RANDMST. Assuming that the graph G is planar, we know by Lemma 6.21 that all fixed component graphs are planar. As a consequence, the number of edges in a fixed component graph is at most three times the number of vertices – more precisely the number of edge sets is at most $3n - 6$, with n denoting the number of fixed components. Since the number of fixed component splits is $k \leq n - 1$, the number of edge sets is at most $3k - 3$.

We modify the cost assignment scheme introduced in the previous section. We assign a new edge set to fixed component splits so that no split is assigned more than 3 edge sets. If an edge set E_S is split, one part of it remains assigned to the split it was assigned to before and for the second part we choose a split as for a new edge set. Since the number of edge sets is always not larger than $3k - 3$, such an assignment exists.

As described in the previous section, the cost of RANDMST, assigned to one edge set, is in expectation $O(\log n)$. So, for planar graphs, we assign at most $O(\log n)$ cost of RANDMST to each fixed component split. Combining this with Corollary 6.18 we can follow Theorem 6.19.

6.5. Weight-Diversified Input Sequences

We can improve on the result obtained for the Restricted ODMST problem on planar graphs by investigating a special class of input graphs and input sequences. We restrict both of them in such a way, that at each point of time the number of edges with a particular weight is not larger than a given constant c. More formally, let $E(w, t)$ be the set of edges with weight w in time step t. Then in a c-weight-diversified sequence we have $|E(w, t)| \leq c$ for each weight w and each time step t.

We will show that RANDMST has a competitive ratio of c on c-weight-diversified graphs.

Theorem 6.22. *For the competitive ratio of algorithm* RandMST *on c-weight-diversified sequences it holds* $\mathcal{R}_{\text{RandMST}} \leq c$.

Once again we reuse the construction of fixed components and edge sets introduced in Section 6.3. Every time a fixed component split S occurs, we assign all created edge sets to the split. Let the split occur on layer w in time step t. By our initial assumption it holds $|E(w,t)| \leq c$.

Consider an edge set E_S created by the split S. Let $x = |E_S \cup E(w,t)|$, i.e. the number of edges in edge set E_S with weight x. The cost of RandMST incurred on this edge set in the remaining execution will be at most x. Therefore, we can bound the cost assigned to F by this edge set to x. Summing up over all edge sets assigned to F, we can bound the total cost assigned to F by c.

Combining this cost assignment scheme with Corollary 6.18 we can follow Theorem 6.22. Note, that for achieving the constant competitive ratio on c-weight-diversified input sequences the randomization of RandMST is not necessary.

6.6. Stochastic Adversary

In this section we will investigate the performance of RandMST against a stochastic adversary for the Restricted ODMST problem. In more detail, we will show that most of the sequences generated by the parameterizable stochastic adversary are c-weight-diversified input sequences.

We assume the following stochastic process. We start with any input graph with all edge weights equal 0. In meta-step t, edge $e \in E$ executes a bernoulli random trial with success probability p_e, independently from other edges. Edge e increases its weight by one in meta-step t if the experiment was successful and leaves its weight as it is otherwise. Since the ODMST problem assumes that at most one edge is increased in a time step, we divide each meta-step into time steps, such that exactly one edge weight is increased in each time step. This defines the input sequence σ. This division can be performed in any way, we do not pose any restrictions on it.

The success probability p_e can be different for each edge $e \in E$ and we denote the vector of probabilities $\pi = (p_e)_{e \in E}$. We only restrict for technical reasons $p_e \leq 1/2$. We denote $m = |E|$ for the input graph $G = (V, E)$.

Theorem 6.23. *For the Restricted ODMST problem and for an input sequence* σ *chosen randomly by the stochastic process the competitive ratio of* RandMST *is constant with high probability.*

Consider all sequences with a length t_{max} generated by the independent stochastic process. We will show that only few of these sequences have $\max_{w \in W \wedge t \in T} |E(w,t)| > c$, with $R = \{t_s, \ldots, t_{max}\}$ and $W \subseteq R$ for an appropriate t_s. We introduce Theorem 6.24 which formally states the fact that the greatest part of most input sequences generated by the described stochastic process has less than c edges with equal weight in every meta-step. We define a technical constant

$$\phi = \max\left\{\max_{e \in E} 1/p_e^8, m^4, m^{\frac{1+c+\alpha}{c/4-3}}\right\} .$$

Theorem 6.24. *For any $t_{\max}$, a set of meta-rounds $T \subseteq \{\phi, \ldots, t_{\max}\}$, and a set of weights $W \subseteq T$ it holds*

$$\mathbf{Pr}\left[\max_{\substack{w \in W \\ t \in R}} |E(w,t)| > c\right] \leq \frac{1}{m^\alpha} .$$

We begin with the following lemma.

Lemma 6.25. *Let $t \geq 1/p_e^8$ and $p_e \leq \frac{1}{2}$ for every $e \in E$. Furthermore, let $t \geq m^{\frac{c+1+\alpha}{c/4-3}}$ and $t \geq m^4$. Then for a fixed meta-step t and a fixed weight $w \leq t$ it holds*

$$\mathbf{Pr}\left[|E(w,t)| > c\right] \leq \frac{1}{m^{\alpha - \log_m \varepsilon}\, t^{2+\varepsilon}} .$$

Proof. Let us look at one specific edge e. The probability that the weight of e is exactly w in meta-step t is

$$\mathbf{Pr}[w^t(e) = w] = \binom{t}{w} p_e^w \left(1 - p_e\right)^{t-w} .$$

Let us consider $\mathbf{Pr}[w^t(e) = w]$ as a function of w. Considering the upper bound on the probability distribution function of the binomial distribution shown in the technical Lemma A.1 (see appendix) we have

$$\max_{w \in \{0,\ldots,t\}} \mathbf{Pr}\left[w^t(e) = w\right] \leq \frac{1}{\sqrt{t \cdot p_e^4}} .$$

As by assumption $t \geq 1/p_e^8$ we have

$$\max_{\substack{w \in \{0,\ldots,t\} \\ e \in E}} \mathbf{Pr}\left[w^t(e) = w\right] \leq \left(\frac{1}{t}\right)^{1/4} . \tag{6.1}$$

Let us turn our attention to the probability $\mathbf{Pr}[|E(w,t)| = i]$ for a fixed meta-step t and a fixed weight w. This probability is defined and upper bounded as follows

$$\begin{aligned}
\mathbf{Pr}[|E(w,t)| = i] &= \sum_{\substack{E' \subseteq E \\ |E'|=i}} \left[\prod_{e \in E'} \mathbf{Pr}[w^t(e) = w] \prod_{e \in E \setminus E'} \Big(1 - \mathbf{Pr}[w^t(e) = w]\Big) \right] \\
&\leq \sum_{\substack{E' \subseteq E \\ |E'|=i}} \left[\Big(\max_{e \in E} \mathbf{Pr}[w^t(e) = w] \Big)^i \Big(1 - \min_{e \in E} \mathbf{Pr}[w^t(e) = w] \Big)^{m-i} \right] \\
&= \binom{m}{i} \Big(\max_{e \in E} \mathbf{Pr}[w^t(e) = w] \Big)^i
\end{aligned}$$

We aim for upper bounding $\mathbf{Pr}[|E(w,t)| = c]$ with $1/(t^3 \cdot m^{1+c+\alpha})$. So,

$$\begin{aligned}
\mathbf{Pr}[|E(w,t)| = c] &= \binom{m}{c} \Big(\max_{e \in E} \mathbf{Pr}\Big[w^t(e) = w\Big] \Big)^c \\
&\leq m^c \Big(\max_{\substack{w \in \{0,\ldots,t\} \\ e \in E}} \mathbf{Pr}\Big[w^t(e) = w\Big] \Big)^c \\
&\leq m^c \cdot \left(\frac{1}{t}\right)^{c/4} . \qquad (6.2)
\end{aligned}$$

From our initial assumptions we have $t \geq m^{\frac{c+1+\alpha}{c/4-3}}$ and thereby

$$\begin{aligned}
m^{c+1+\alpha} &\leq t^{c/4-3} \\
m^c \cdot \left(\frac{1}{t}\right)^{c/4} &\leq \frac{1}{t^3 \cdot m^{1+\alpha}} .
\end{aligned}$$

Therefore we have

$$\mathbf{Pr}[|E(w,t)| = c] \leq \frac{1}{t^3 \cdot m^{1+\alpha}} .$$

Now we want to upper bound the probability that the number of edges with weight w in a meta-step t is greater than c

$$\begin{aligned}
\mathbf{Pr}[|E(w,t)| \geq c] &= \sum_{i=c}^{m} \mathbf{Pr}[|E(w,t)| = i] \leq m \cdot \max_{i=c,\ldots,m} \mathbf{Pr}[|E(w,t)| = i] \\
&\leq m \cdot \max_{i=c,\ldots,m} \mathbf{Pr}[|E(w,t)| = i] \qquad (6.3) \\
&\leq m \cdot \max_{i=c,\ldots,m} m^i \cdot \left(\frac{1}{t}\right)^{i/4} . \qquad (6.4)
\end{aligned}$$

By assumption $t \geq m^4$ and therefore $(1/t)^{1/4} \leq 1/m$. This implies $m^i (1/t)^{i/4}$ is largest for $i = c$. Thus,

$$\mathbf{Pr}[|E(w,t)| \geq c] \leq \max_{i=c,\ldots,m} m^i (1/t)^{i/4} \leq m^c (1/t)^{c/4} \leq \frac{1}{t^3 \cdot m^\alpha} .$$

■

With the help of the previous lemma we will prove Theorem 6.24. Using the result from the former lemma we can approximate the following probability, for $T \subseteq \{\phi, \ldots, t_{max}\}$ and $W \subseteq T$

$$\begin{aligned}
\mathbf{Pr}\left[\max_{w\in W \wedge t\in T} |E(w,t)| \geq c\right] &\leq \mathbf{Pr}\left[\bigcup_{w\in W\wedge t\in T} |E(w,t)| \geq c\right] \\
&\leq \sum_{w\in W\wedge t\in T\wedge w\leq t} \mathbf{Pr}[|E(w,t)| \geq c] \\
&\leq \sum_{w\in W\wedge t\in T\wedge w\leq t} \frac{1}{t^3\cdot m^\alpha} \leq \sum_{t\in T} t\frac{1}{t^3\cdot m^\alpha} \\
&\leq \frac{1}{m^\alpha}\sum_{t\in T}\frac{1}{t^2} \\
&\leq \frac{1}{m^\alpha}\,.
\end{aligned}$$

The last inequality holds because $\sum_{t\in T}\frac{1}{t^2} \leq 1$. This concludes the proof of Theorem 6.24.

Splitting meta-steps. The following lemma shows that the limit on $|E(w,t)|$ established by Theorem 6.24 can be applied to steps, created by splitting meta-steps.

Lemma 6.26. *Assume that $|E(w,t)| \leq c$ for all $t \in T$ and $w \in W \subseteq T$. Then after splitting the set of meta-steps T into a set of steps T' it holds $|E(w,t)| \leq 2c$ for all $t \in T'$ and $w \in W' \subseteq T'$.*

Proof. Let t_m be a meta-step and assume there exists a step $t \in T'$ created from t_m and a weight w such that $|E(w,t')| > 2c$. Each edge which has weight w in time step t had either weight w or $w-1$ at the beginning of meta-step t_m. As $|E(w,t')| > 2c$, it must hold $|E(w,t_m-1)| > c$ or $|E(w-1,t_m-1)| > c$, leading to a contradiction. ■

Competitive ratio. We divide the input sequence σ in two parts and consider the competitive ratio of RANDMST on both parts as on separate sequences. The first part σ_1 reaches from the first time step to ϕ. The second part σ_2 consists of the rest of σ. We choose the initial tree for the input sequence σ_2 as the last $\mathcal{M}_{\text{ALG}}$ computed by RANDMST on σ_1. Then the execution of RANDMST on the whole sequence σ is the same as the execution of RANDMST on σ_1 first and σ_2 afterward. Therefore, $C_{\text{ALG}}(\sigma_1) + C_{\text{ALG}}(\sigma_2) = C_{\text{ALG}}(\sigma)$.

Since σ_1 has a length of ϕ it holds $C_{\text{ALG}}(\sigma_1) \leq \phi$. By Theorem 6.24 and Lemma 6.26 it holds $|E(w,t)| \leq 2c$ for all time steps in σ_2 with high probability. If that event occurred, then Theorem 6.22 can be applied to bound RandMST's cost on σ_2, so that

$$C_{\text{ALG}}(\sigma_2) \leq 2c \cdot C_{\text{OPT}}(\sigma_2) + k\,, \tag{6.5}$$

where k is an appropriate constant independent of the input sequence and the definition of the stochastic process. Altogether we obtain

$$\mathbf{Pr}\left[C_{\text{ALG}}(\sigma) \leq 2c \cdot C_{\text{OPT}}(\sigma) + \phi + k\right] \geq 1 - 1/m^{\alpha}\,, \tag{6.6}$$

which implies Theorem 6.23.

6.7. Hardness of Approximated ODMST Problem

In the ODMST problem we have forced the online algorithm to always maintain a *minimum* spanning tree. One might be tempted to relax this requirement and allow the algorithm to maintain a c-approximation of a MST and thereby (possibly) reduce the cost necessary to maintain the MST.

To model this scenario, we define the c-apODMST problem. Formally, in the c-apODMST problem the online algorithm is allowed to output a c-approximation of a MST in each round. On the other hand, we still expect from the optimal offline algorithm that it outputs a perfect minimum spanning tree in each round. Thereby, we allow the online algorithm more freedom in comparison to the optimum offline algorithm. We keep the cost measures for both algorithms as in the original ODMST algorithm and stick to the notion of competitiveness given earlier.

Unfortunately, even with this relaxation, both lower bounds shown for the original ODMST problem hold also for c-apODMST. We briefly describe how to modify the construction of the lower bounds to make account for the greater possibilities of the online algorithm. For the construction we once again utilize the graph used for the deterministic lower bound on the ODMST problem (see Section 6.1). Recall, that the vertices of G are split thereby in two sets V_1 and V_2, s.t. $n/2 = |V_1| = |V_2|$. All edges connecting vertices of one of those sets have weight 1 and all edges crossing the boundary have initially weight n. Therefore, the weight of any MST for this graph is $2\cdot(n/2-1)+n = 2n-2$.

In the construction of input sequences for both lower bounds we have been gradually increasing the weight of edges from E_C. Let σ be the original input sequence, used in Section 6.1. basing on σ, we create a new sequence σ_{ap}. Let t be a time step in σ, such that the weight of edge e is increased by one. Increasing this edge weight caused the algorithm Alg solving the ODMST problem to abandon this edge and use another one

for $\mathcal{M}_{\text{ALG}}$. In order to achieve the same for the c-apODMST problem, we have to increase the weight of e by more than one. Up to time step t, the weight of $\mathcal{M}_{\text{ALG}_{\text{ap}}}$, the minimum spanning tree constructed by the algorithm ALG_{ap}, is equal to $n-2+w^t(e)$. Therefore, we have to increase the weight of e to $c \cdot (n-2+w^t(e)) - n + 3)$. After such an increase, the weight of a spanning tree including e is larger than c times the weight of the minimum spanning tree, and ALG_{ap} is forced to remove e from $\mathcal{M}_{\text{ALG}_{\text{ap}}}$.

Therefore, if σ forces any algorithm for the ODMST problem to have a competitive ratio of C, then σ_{ap} does the same for the any algorithm solving the c-apODMST problem. We can conclude, that even with this relaxation, the c-apODMST problem remains hard.

CHAPTER 7

Managing Spare Relays

In this chapter we are considering the problem of an appropriate management of relays which are not actively used. Even if relays are not used at a given moment, they should be on standby for being used in one of the various communication chains in the robot graph. The key problem occurs when at a certain location in the robot graph the demand for relays increases – the question is then which of the spare relays to use. We want to minimize the total energy used for traveling by relays. An intuitive way to answer an increasing demand for relays would be to use the relay which is closest to the source of demand. Unfortunately, this is not the best way to go, sometimes it makes sense to fetch a relay from a more distant location.

Basing on this description one can think of using this framework for handling any resources, not necessarily relays. Therefore we formulate our problem as the *k*-resource problem.

The *k*-resource problem is an extension of the well-known *k*-server problem. It is based on a finite metric space $(\mathcal{X}, \xi)$. The metric space is equivalent to a graph and therefore we will use both notations interchangeably within this chapter, depending on which is more convenient for a specific purpose. Whenever we speak of a graph, we mean a complete graph with nodes corresponding to points of $\mathcal{X}$, and with edge weights given by function ξ.

In each point of $\mathcal{X}$ we have some demand for resource units. The demands are arbitrary integers, but their sum is guaranteed to be at most some fixed integer *k*. We investigate the *k*-resource problem in the framework of competitive analysis, therefore the concept of time is necessary. Time is slotted and divided into time steps. In each time step, the algorithm solving the *k*-resource problem is revealed one item of the input sequence, which contains changes of the demand: in every time step, at exactly one place the demand decreases or increases by 1. At the end of any time step, the algorithm has to *cover* all the demands by its resource units, which means that the number of resources

available at any node has to be at least the demand at this node. In order to do that, the algorithm may move the units between the points of $\mathcal{X}$, paying the sum of distances traveled by the units.

More formally, let the *resource vector* X_{t-1} denote the algorithm's distribution of resources to points in the metric space at the end of the time step $t-1$. We assign all movement of resources by the algorithm to a particular time step, therefore the distribution of resources at the end of time step $t-1$ is equal to that at the beginning of time step t. In step t, the algorithm is given a *demand vector* d_t, which describes the demand for resources in all points of the metric. We allow only input sequences, for which $\sum_{i=1}^{n} |d_{t-1}[i] - d_t[i]| = 1$ and $|d_t| \leq k$ for all time steps t. This restricts the change of demand between two consecutive time steps to consist of one resource unit at one point of the metric. Furthermore, the total demand may never exceed k. We assume that at the very beginning of the input sequence, X_0 is any but fixed and $|d_0| = 0$, meaning there is no demand.

During the time step, an algorithm has to move resources between metric points, so that at the end of the time step $X_t \geq d_t$, where X_t is the new resource vector. For each resource unit moved from point i to j the cost of the algorithm is equal to the distance $\xi(i, j)$ between these two points. The total cost of the algorithm during a time step is the sum of the cost of all movements performed in this time step.

Metrics

We have defined the k-resource problem for arbitrary metrics. We will show later by a simple argument, that creating an algorithm with competitive ratio $o(k)$ on any metric for the k-resource problem is considered difficult in literature. Therefore, we will concentrate on two special types of finite metrics.

Uniform metric space. In the *uniform* metric space, the distances between two different points of the space are equal to 1. Therefore moving a resource unit between two points in the metric always incurs unit cost to the algorithm. We will see that reasonable algorithms for uniform metric spaces move at most one resource unit per time step and therefore their cost for a time step is either none or 1.

Generally, one can allow for small differences between the distances of points in the uniform metric spaces. If the ratio between the smallest and largest distance is bounded by c, then the competitive ratio of an algorithm, designed for a true uniform metric space, is only by the factor c worse if executed on such a metric space. Therefore, one can define a uniform metric space so that the distances are all $\Theta(1)$, causing no change to the competitive ratio which is usually stated in the $O(\cdot)$-notation.

Decomposable metric spaces. A metric space $(\mathcal{X}, \xi)$ is *decomposable* if it can be partitioned into a set Λ of disjoint components. The distance between components has to be relatively large comparing to the maximal distances between points within individual components. As we won't consider general decomposable metric spaces in our work, we skip a formal definition of a decomposable metric space for a moment and get back to it in the Related Work section.

Uniformly decomposable metric spaces. A uniformly decomposable metric space is a decomposable metric space fulfilling additional restrictions. Each of the components from Λ has to be a uniform metric space, i.e. the distance between two points $i, j \in \lambda$ in the component $\lambda \in \Lambda$ have to be $\xi(i, j) = 1$. Furthermore, the components have to be pairwise separated by Γ, so that for any two points $i \in \lambda_1, j \in \lambda_2$ from distinct components $\lambda_1, \lambda_2 \in \Lambda$, $\lambda_1 \neq \lambda_2$ it holds $\xi(i, j) = \Theta(\Gamma)$. The value Γ will be denoted as the *separation* between components.

Uniformly decomposable metrics are a natural extension of the uniform metric, capable of capturing some aspects of locality. Intuitively, the requirement for all intra-component distances to be equal to Γ is that we are going to treat these connections as a uniform metric on a high-level, such that each component is considered like a point.

Exactly as for the uniform metrics we may relax the requirement on the distances to $\Theta(\Gamma)$ instead of Γ.

Dynamic-Resource Problem

In order to solve the k-resource problem on uniformly decomposable metric spaces we want to reuse algorithms which solve the same problem on uniform metric space. Unfortunately, a solution of the k-resource problem for uniform metric spaces is insufficient for this purpose. Therefore, we define a slightly more general problem, the dynamic-resource (DR) problem. An algorithm solving the DR problem on uniform metric spaces can be later used for solving the k-resource problem on uniformly decomposable metric spaces.

In the dynamic-resource problem the number of resources available at the beginning of each step t may change. If it increases, the algorithm may choose where to place the resources; if it decreases, the algorithm may choose which units are removed. We also generalize the competitive analysis for this problem: the number of units available to the online algorithm in step t (denoted by k_t) is not necessarily equal to the number of units at optimal offline algorithm's disposal (k_t^{OPT}). Naturally, in each step $|d_t| \leq k_t$ and $|d_t| \leq k_t^{\mathrm{OPT}}$.

In general, the DR problem seems to be much harder than k-resource, and we do not aim at constructing competitive algorithms for it. Instead, we construct algorithms which perform well on some instances of DR, in particular when $k_t \geq k_t^{\mathrm{OPT}}$, k_t^{OPT} is non-increasing, and k_t is non-decreasing. Note that the setting $k_t \equiv k_t^{\mathrm{OPT}} \equiv k$ corresponds to the original k-resource problem.

Resource augmentation

Solving the k-resource problem on uniformly decomposable metric spaces will depend on c-resource augmentation. That means that we will allow our algorithm to use $(1+c)\cdot k$ resources to solve its task, whereas the offline schedule will have to fulfill the same task using only k resource items. Obviously the input sequence will be still limited so that $|d_t| \leq k$, otherwise the offline schedule couldn't solve the problem at all. We call this version of the k-resource problem the *c-resource augmented* version.

Results and Outline

We start our investigation of the k-resource problem by presenting its relation to the k-server problem. This yields lower bounds on the competitiveness of any algorithm solving k-resource, as shown in Section 7.1. The main results of the paper are divided into two parts. In Section 7.2, we consider the k-resource problem on uniform metric spaces consisting of n points. We present a deterministic and a randomized algorithm, both achieving asymptotically optimal competitive ratios: $O(\min\{n,k\})$ and $O(\log(\min\{n,k\}))$, respectively. In fact, these algorithms are able to solve some instances of dynamic-resource problem.

The results above allow us to solve the k-resource problem on uniformly decomposable metric spaces with $\Gamma \geq k$. The key issue for this kind of metrics is to use resources available locally in the component and transport new ones from other components only rarely. In Section 7.3 we give a randomized algorithm with a competitive ratio $(1 + 1/c) \cdot O(\log(\min\{n,k\}))$, where n is the size of the largest component of the metric space and c denotes the resource augmentation factor (see Theorem 7.21). By the relations between the k-resource and k-server problems, the same result holds for k-server. The competitiveness does not depend on the number of components in the decomposition.

Related Work

The k-server problem, introduced in [MMS90], is defined on a metric space $(\mathcal{X}, \xi)$, where $\xi(\cdot,\cdot)$ is a function measuring the distance between two points of space $\mathcal{X}$. An input consists of sequence of requests to points from $\mathcal{X}$. Upon seeing a request to point $x \in \mathcal{X}$, an algorithm has to move its servers, so that at least one server is placed at x. The costs

of transports are defined by the distance function ξ. The goal of an algorithm solving the k-server problem is to minimize the total distance traveled by its servers in the runtime.

The setting of the k-server problem is powerful enough to model many applications in computer science, as well as in many other aspects of everyday life. On the other hand, the model has also some deficiencies. The k-resource problem, a very natural extension defined in this paper, has the advantage of dealing additionally with two of those: we allow requests to have different durations (whereas in k-server a request is regarded as processed after one unit time step) and we allow to request more than one server to a point of the metric. An application example for the k-resource could be a robotic scenario, where tasks appearing dynamically in a terrain impose certain demands (for some periods) on the number of robots that have to be sent to handle them.

Results on k-server. The k-server problem has been prominent in the research on online problems from the advent of competitive analysis. Currently the best algorithm for general metrics, WFA [KP95], is $(2k-1)$-competitive, which nearly matches the lower bound of k by [MMS90]. Although this result gives a very important insight into this and related problems, the achieved competitiveness is hardly acceptable for most applications.

Thus, much effort was put in the search for randomized algorithms which could perform better. The results for general metric spaces are unsatisfactory: whereas there exists a lower bound on the competitive ratio of $\Omega(\log k/\log\log k)$ [BBM01, BLMN03], currently there is no randomized algorithm better than k-competitive. There are, however, several results for specific metric spaces that break the $o(k)$ barrier. A classical result [FKL+91] shows a randomized $O(\log k)$-competitive algorithm for the uniform metric space. Recently, results on other metric spaces were obtained: the work by Csaba and Lodha [CL06] provides an $O(n^{2/3}\log n)$-competitive algorithm for n equally spaced points placed on a line; Bartal and Mendel [BM05] generalized this work for growth-rate bounded graphs. Seiden [Sei01] has related the k-server problem on decomposable metric spaces to unfair metrical task systems.

Our result for k-resource on uniformly decomposable metric spaces is one of the few, which relies on resource augmentation. Another successful applications are due to Young [You02] for the weighted caching problem and Sleator and Tarjan [ST85] for paging.

Decomposition theorem in k-server. We will now give a formal defin restricted model of uniformly decomposable metric spaces.

Partition the point set $\mathcal{X}$ into a set of components $\Lambda = \{\lambda_1, \ldots, \lambda_k\}$. Let $\mathrm{dia}(\lambda) = \sup_{i,j\in\lambda} \xi(i,j)$ be the diameter of component λ and $\mathrm{dia}(\Lambda) = \max_{\lambda\in\Lambda} \mathrm{dia}(\lambda)$ be the maximum diameter over all components. The distance between two components is defined

as

$$\xi(\lambda_1, \lambda_2) = \max_{i \in \lambda_1, j \in \lambda_2} \xi(i, j) .$$

Then the separation of the components is equal to

$$\tilde{\Gamma} = \min_{\lambda_1, \lambda_2 \in \Lambda} \frac{\xi(\lambda_1, \lambda_2)}{\mathrm{dia}(\Lambda)}$$

The results provided by Seiden [Sei01] assume only that $\tilde{\Gamma}$ is sufficiently large. This implies, that the distances between components are large in comparison to the maximum distances within components. There are no restrictions implying that the distances between components are similar. The presented algorithms provide an algorithm with expected competitive ratio of approximately $O(z \log^2(k+z))$ where $z = |\Lambda|$ (for clarity of presentation we have neglected a few lower-order terms in the competitive ratio). The solution requires a separation of at least $\tilde{\Gamma} > 2k$.

Note that achieving an expected competitive ratio of $o(|\Lambda|)$ for the setting of decomposable metric spaces would solve the k-server with expected competitive ratio $o(|\Lambda|)$. Though there exists no lower bound showing this is infeasible, this is considered a very hard task. Therefore, looking for a result with competitive ratio sublinear in z (or even independent from z) we have restricted the metric slightly more, defining Γ in a stricter way than $\tilde{\Gamma}$ and requiring each component to be a uniform metric space.

Model Restrictions

The described model has certain limitations. The most important is that we allow changes in the network to happen only at a slow rate, i.e. the demand changes at only one point and only by one per time step. A similar restriction has been already imposed on the model of the Online Dynamic Minimum Spanning Tree problem and discussed in Chapter 6.

7.1. Lower Bounds

It is quite straightforward that the k-resource problem is capable of modeling k-server instances. Moreover, the ability to solve the k-resource problem implies the ability to solve the k'-server problem, for $k' \leq k$, as stated in the lemma below.

Lemma 7.1. *Fix any metric space $(\mathcal{X}, \xi)$ and integers $k' \leq k$. If there exists an α-competitive (randomized) algorithm A for the k-resource problem on $(\mathcal{X}, \xi)$, then there exists an α-competitive (randomized) algorithm A' for the k'-server problem on $(\mathcal{X}, \xi)$.*

Proof. Consider any input σ_{serv} for k'-server problem; we show how to construct an input σ_{res} for the k-resource problem. We choose any node $v \in \mathcal{X}$, and at the beginning of σ_{res} we increase the demand at v to $k - k'$. Then every round of the original instance is translated into two rounds for the k-resource problem: a request at x is converted to an increase of resource demand at x by 1 in the first round and decrease back by 1 in the second round. The algorithm A' runs algorithm A on σ_{res} and returns A's movements of units as its own solution.

Note that A has only k' *free* units; the other ones are bound to v for the whole runtime. Thus, the movements of these units correspond to a feasible solution for k'-server problem. Moreover, the costs of these two solutions coincide: $C_{A'}(\sigma_{\text{serv}}) = C_A(\sigma_{\text{res}}) - \beta'$, where β' is the cost of initial movement of $k - k'$ units to v.

Let β be the maximum possible cost of sending $k - k'$ units from any points of $\mathcal{X}$ to v. Note that β depends only on the structure of the metric space and not on the input sequence. Therefore, any optimal solution for σ_{serv} yields a solution S for σ_{res} with the cost at most β greater. Thus, $C_{\text{OPT}}(\sigma_{\text{res}}) \le C_S(\sigma_{\text{res}}) \le C_{\text{OPT}}(\sigma_{\text{serv}}) + \beta$.

Summing up, since A is α-competitive, there exists a constant ρ, such that $C_A(\sigma_{\text{res}}) \le \alpha \cdot C_{\text{OPT}}(\sigma_{\text{res}}) + \rho$. Therefore,

$$C_{A'}(\sigma_{\text{serv}}) \le C_A(\sigma_{\text{res}}) \le \alpha \cdot C_{\text{OPT}}(\sigma_{\text{res}}) + \rho \le \alpha \cdot C_{\text{OPT}}(\sigma_{\text{serv}}) + (\alpha \cdot \beta + \rho) \ ,$$

which proves the α-competitiveness of A'.

The result for the randomized case follows when we replace C_A and $C_{A'}$ with their expected values. ■

By means of Lemma 7.1, we may apply known lower bounds for k-server (which are: k for a deterministic scenario [MMS90], $\Omega(\log k / \log\log k)$ for a randomized one [BBM01, BLMN03], and $H_k = \Theta(\log k)$ for a randomized one in uniform spaces [FKL+91]) to achieve the bounds on k-resource problem. Since the bounds on k-server hold if k is smaller than the number of points of the metric space, we relate the k-resource problem to a $(\min\{k, n-1\})$-server problem, where n is the number of points in $\mathcal{X}$. If $\mathcal{X}$ is an infinite space, then $\min\{k, n-1\} = k$.

Corollary 7.2. *Fix any space $(\mathcal{X}, \xi)$ consisting of n points, any integer k, and any α-competitive algorithm A for the k-resource problem. Let $f = \min\{k, n-1\}$. If A is deterministic, then $\alpha \ge f$; if A is randomized, then $\alpha = \Omega(\log f / \log\log f)$. Moreover, if $(\mathcal{X}, \xi)$ is a uniform metric, then for a randomized A, $\alpha \ge H_f$, where $H_i = \sum_{j=1}^{i} 1/j = \Theta(\log i)$, the i-th harmonic number.*

7.2. Uniform metric spaces

In this section we investigate the k-resource and dynamic-resource problems on a uniform graph of $n \geq 2$ nodes. First, we present some definitions and notations. Then, we specify a very general class of *phase-based algorithms* and prove crucial properties about them. It appears that any (deterministic) algorithm from this class is $\tilde{O}(\min\{n,k\})$-competitive. Finally, we show that by carefully using randomness, we are able to substantially reduce this factor to $\tilde{O}(\log(\min\{n,k\}))$. By Corollary 7.2, our results are asymptotically optimal.

Preliminaries. Formally, by a *vector* we understand a vector of n integers. For any vector A, let $|A| = \sum_{i=1}^{n} A[i]$. We say that a vector A covers B (we write $A \geq B$) if $A[i] \geq B[i]$ for all $1 \leq i \leq n$. For any two vectors A and B we define the *distance* between them as $\delta(A,B) := \frac{1}{2} \cdot \sum_{i=1}^{n} |A[i] - B[i]|$.

Recall that *at the beginning* of each step t, the resource vector X_{t-1} defines the distribution of resources to graph nodes. The demand vector d_t must be satisfied by the algorithm at the end of time step t, by moving resources appropriately so that $X_t \geq d_t$. It is straightforward to see that for uniform metrics the algorithm has to pay at least $\delta(X_{t-1}, X_t)$ for changing the resources from X_{t-1} to X_t, and it is possible to schedule the movements in such a way that the cost is equal to $\delta(X_{t-1}, X_t)$. In the following we omit subscript t if it can be deduced from the context.

Division of input into phases. First, we show how to deterministically partition the whole input sequence σ into phases; each phase consist of several time steps. The procedure is described in Algorithm 3. In each phase we track the maximum demand occurring at any node, and we store this information in vector D_t. Within one phase, D is non-decreasing and since the change of d is restricted, only one entry of D can change, and only by 1. At the beginning of the phase P_q we store the current number of units, k_t, in the variable κ_q. Note that a phase P_q may end because either $k_t < \kappa_q$ or $|D_t| > \kappa_q$. If the former condition holds, we call such P_q *unfair*, otherwise P_q is *fair*. Note that any k-resource instance consists only of fair phases. As a byproduct, we get that within a phase $|d_t| \leq |D_t| \leq \kappa_q$. Note that the partitioning into phases depends only on the input sequence and can be computed online.

7.2.1. Phase-based algorithms for the DR problem

In this section we construct a class of *phase-based* algorithms. Although the algorithms will always produce a feasible solution for the DR problem, we prove their competitiveness in fair phases only. This immediately yields their competitiveness on k-resource

Algorithm 3 Division of the input sequence into phases

```
1: procedure StartPhase(q, t)
2:     P_q ← {t};  κ_q ← k_t;  D_t ← d_t
3: end procedure

4: q ← 1; StartPhase (q, 1)                          ▷ P_q = current phase
5: for t ← 2 to |σ| do
6:     for all node i do
7:         D_t[i] ← max_{t'∈P_q} d_{t'}[i]
8:     end for
9:     if k_t < κ_q or |D_t| > κ_q then
10:        q ← q + 1;  StartPhase (q, t)              ▷ new phase starts now
11:    else
12:        P_q ← P_q ∪ {t}
13:    end if
14: end for
15: return (P_1, P_2, ..., P_q)
```

instances.

The proof consists of two parts: in the first one we relate the cost of the optimal offline algorithm in each two consecutive phases P_{q-1} and P_q to the difference of the values of D at the end of these phases; in the second one we relate the cost of our algorithm to the same amount.

Algorithm construction. A phase-based algorithm PB bases its choices solely on the contents of vector D, namely it always tries to maintain an invariant $X \geq D$. This simplifies the analysis, as we may assume that the adversary manipulates vector D directly, instead of changing the demands. Moreover, in phase P_q, the algorithm uses only κ_q units, and we explicitly assume it in the analysis, i.e. $|X| = \kappa_q$. The surplus of units is left untouched by the algorithm and is stored in a fixed node v_S. If the real number of units decreases, the surplus decreases, and if surplus diminishes below zero, it implies the end of the phase because of the condition $k_t < \kappa_q$.

First, we describe the behavior of PB in the first step of a phase P_q. If the preceding phase was fair, then $\kappa_q \geq \kappa_{q-1}$. If this inequality is strict, then the algorithm has to increase the number of resources it is actually using. We simply assume that at the very beginning of P_q, the vector X increases by the surplus units, so that $|X| = \kappa_q$. This introduces no additional cost to the algorithm. If the preceding phase was unfair, then the situation is a little bit more complicated, because $\kappa_q < \kappa_{q-1}$, and the algorithm has to

remove some units. In such case, in the first step a *reorganization* is performed, i.e. PB moves all the units, so that the demand is fulfilled exactly: $X[i] = d[i]$ for each node i but v_S, where all the remaining units are stored.

To precisely describe the behavior of PB within a phase, we divide the nodes into three groups. A node i is called *saturated* when $X[i] = D[i]$, *unsaturated* when $X[i] > D[i]$, and *infringing* when $X[i] < D[i]$. It usually makes a difference, when these conditions are checked: *before* or *after* the algorithm's decision in step t. In the former case, D_t is compared with X_{t-1}, and in the latter with X_t.

Phase-based algorithms are *lazy* (as introduced earlier for the k-server problem in [MMS90]). It means that if the invariant holds at the beginning of time step t, i.e. if $X_{t-1} \geq D_t$, the algorithm does nothing ($X_t = X_{t-1}$). Otherwise, note that X_{t-1} covers D_{t-1} and since D may change only at one node, say j, there is at most one infringing node j. The algorithm may not move any of the units from saturated nodes without further violating the invariant, and therefore it considers the set of unsaturated nodes $\mathcal{U}_t = \{i \in V : X_{t-1}[i] > D_t[i]\}$. Since throughout the phase $|D_t| \leq \kappa_q = |X_t|$ and there exists one infringing node, $\mathcal{U}_t$ is non-empty. PB chooses now one node $i \in \mathcal{U}_t$, according to some rule called $\mathcal{U}$*-rule,* and moves one unit from node i to j (we write that j *borrows* one unit from i). Note that PB is not fully defined, as we have some freedom in choosing the $\mathcal{U}$-rule. It appears that we may show some properties of PB and the bounds on its performance which hold for any $\mathcal{U}$-rule.

Basic observations. We start with an easy observation on the structure of $\mathcal{U}_t$ sets and node saturation process.

Lemma 7.3. *For any two consecutive steps t and $t+1$ from one phase, $\mathcal{U}_{t+1} \subseteq \mathcal{U}_t$. Additionally, at the very end of a fair phase all nodes are saturated.*

Proof. For the first part of the lemma, it suffices to show that within a phase it is impossible for a node to change its state to unsaturated. Assume the contrary, i.e. that a node i changed its state into unsaturated. Since D is non-decreasing, it can only happen if the number of resources at i increased, i.e. $X_{t+1}[i] > X_t[i]$. But the algorithm PB increases the number of resources only when a node is infringing. Then, after the increase, the node becomes saturated (and not unsaturated).

For proving the second part of the lemma, we fix any fair phase P_q. At the end of each step t from P_q, we have $X_t \geq D_t$ and $|X_t| = \kappa_q$. Since P_q is fair, at the end of its last step $|D| = \kappa_q$, and thus it holds that $X = D$, i.e. all nodes are saturated. ■

Now fix any phase P_q. For the sake of analysis, if a reorganization of X occurs at the end of the first step of P_q, we assume that it is performed at the very beginning of P_q, before the demands are presented to the algorithm in the first step. This assumption simplifies the notations below.

Let F_q be the resource vector X at the very beginning of P_q (but after the reorganization step, if any) and let F'_q be the resource vector at the very end of P_q. Obviously, it holds that $|F_q| = |F'_q| = \kappa_q$. If at some node we need a resource unit, we borrow it from a node which has too many units compared to its current demand. If we always borrowed from a "correct" node (one that will not need this unit anymore in this phase), we would pay exactly $\delta(F_q, F'_q)$.

Lower bound on the optimal algorithm. It appears that *on fair phases* no algorithm which uses few resources can asymptotically beat the cost of $\delta(F_q, F'_q)$, as stated in the following lemma.

Lemma 7.4. *For any two consecutive fair phases P_{q-1} and P_q, any algorithm for the DR problem, which has at most* $\min\{\kappa_{q-1}, \kappa_q\}$ *units in these phases, has to pay at least $\delta(F_q, F'_q)/2$.*

To prove this lemma, we first introduce a new notation. For any phase P_q let τ_q denote the value of vector D at the end of the phase. Note that for a fair phase P_q, $|\tau_q| = \kappa_q$. Additionally, if P_q is a fair phase, then $F'_q = \tau_q$; if P_{q-1} is a fair phase, then $F_q \geq \tau_{q-1}$. It is possible to show the following technical lemma.

Lemma 7.5. $\delta(\tau_{q-1}, \tau_q) \geq \delta(F_q, F'_q)/2$ *for any two consecutive fair phases P_{q-1} and P_q.*

Proof. Recall that since P_{q-1} and P_q are fair phases, $F_q \geq \tau_{q-1}$ and $F'_q = \tau_q$. Additionally,

$|F_q| = |F'_q|$. Therefore

$$\begin{aligned}
\delta(F_q, F'_q) &= \sum_{i:F'_q[i]>F_q[i]} (F'_q[i] - F_q[i]) \\
&= \sum_{i:\tau_q[i]>F_q[i]} (\tau_q[i] - F_q[i]) \\
&\leq \sum_{i:\tau_q[i]>\tau_{q-1}[i]} (\tau_q[i] - \tau_{q-1}[i]) \\
&\leq \sum_i |\tau_q[i] - \tau_{q-1}[i]| \\
&= 2 \cdot \delta(\tau_{q-1}, \tau_q) \ .
\end{aligned}$$

■

Thus, it is sufficient to prove that any algorithm has to pay at least $\delta(\tau_{q-1}, \tau_q)$ for two phases P_{q-1} and P_q. To give the reader more insight in the structure of the lower bound, we consider the following thought experiment. Assume for a while that we are dealing with the k-resource problem, i.e. $k_t \equiv k_t^{\mathrm{OPT}} \equiv k$. Consider the following strategy OFF for an offline algorithm: at the beginning of phase P_q, set the resource vector to τ_q and do not change it throughout the whole phase. Of course, such a resource vector is valid for all the demands occurring in the phase, and thus inside a phase OFF pays nothing. A drawback of this algorithm is that the changing between τ_{q-1} and τ_q might be expensive. The proof of lemma 7.4 that this "switching" cost is unavoidable by *any* algorithm, even the optimal offline one. Moreover, this lower bound holds not only for k-resource instances, but also for fair phases of DR instances.

Proof of Lemma 7.4. Let $\kappa = \min\{\kappa_{q-1}, \kappa_q\}$. Let $f_i = \max\{\tau_{q-1}[i], \tau_q[i]\}$, i.e. f_i is the maximum demand occurring at node i during the given two phases. Consider the following thought experiment: we have $\sum_i f_i$ slots, at most κ slots are occupied by units at the beginning. We want each slot to be occupied at some moment of time, as this is a condition necessary to have f_i resources at node i at some time step. To each of $\sum_i f_i - \kappa$ slots, which are initially empty, we have to move a unit, and thus the total amount paid is at least $\sum_i f_i - \kappa$. It suffices to bound this term by $\delta(\tau_{q-1}, \tau_q)$. By the definition of δ, we have

$$2 \cdot \delta(\tau_{q-1}, \tau_q) = \sum_{i:\tau_{q-1}[i]>\tau_q[i]} (\tau_{q-1}[i] - \tau_q[i]) + \sum_{i:\tau_q[i]>\tau_{q-1}[i]} (\tau_q[i] - \tau_{q-1}[i]) \ .$$

We assume that the first sum is greater; the other case is symmetric. Then we obtain

$$\begin{aligned}
\delta(\tau_{q-1}, \tau_q) &\leq \sum_{i:\tau_{q-1}[i]>\tau_q[i]} (\tau_{q-1}[i] - \tau_q[i]) \\
&= \sum_{i:\tau_{q-1}[i]>\tau_q[i]} (\tau_{q-1}[i] - \tau_q[i]) + \sum_{i:\tau_{q-1}[i]\leq\tau_q[i]} (\tau_q[i] - \tau_q[i]) \\
&= \sum_i \max\{\tau_{q-1}[i], \tau_q[i]\} - \sum_i \tau_q[i] \\
&= \sum_i f_i - \kappa_q \\
&\leq \sum_i f_i - \kappa \ ,
\end{aligned}$$

which finishes the proof. ■

Note that, generally, the lemma above does not hold for unfair phases. In particular τ vectors might be small there, and thus it might be possible for an algorithm to have a resource vector which is "good" for both phases, i.e. does not force the algorithm to move units.

Tokens and cost in one phase. In an online setting it is usually not possible to borrow units in an optimal way. If — to fulfill the demand — node i borrows from some node j, which will need this unit in the future, we only get rid of the deficiency for a while and when the demand at node j increases appropriately, it will have to borrow from some other node.

To analyze the performance of our algorithm we introduce a concept of *tokens*. One may think of a token as a kind of currency used to pay for units sent from remote nodes. The phase begins with no tokens in the graph; at the end of a phase all tokens are removed. Whenever node i borrows a unit from node j it has to pay j, i.e. give j a token in exchange. If i has no tokens, then prior to the exchange it has to *produce* one. In effect, the cost paid by the algorithm in one phase is equal to the number of node changes of all the tokens produced in this phase. We denote the number of tokens at node i at the end of time step t by $T_t[i]$. On the basis of the description above, one can deduce that only at most $\delta(F_q, F'_q)$ tokens are produced during any phase P_q.

To prove this, we first formulate the relation between the D, X and T, and the bound on the number of tokens produced.

Lemma 7.6. *Fix any phase P_q. For any time step t from P_q and any node i, it holds that $X_t[i] + T_t[i] \geq F_q[i]$. Moreover, if $X_t[i] > D_t[i]$ or $T_t[i] > 0$, then $X_t[i] + T_t[i] = F_q[i]$.*

Proof. We prove the lemma by induction on the number of steps within phase P_q. At the beginning of P_q, there are no tokens at node i and $X_t[i] = F_q[i]$, so the induction basis trivially holds. Assume that the conditions of the lemma hold for time step t. For the vectors X or T to change in time step $t+1$, D has to increase at some node i, so that i becomes an infringing node and has to borrow one unit from a node j. We consider the lemma condition separately at node i and j.

1. If there were some tokens at i in time step t, then a token was sent in exchange for a unit, and thus $X_{t+1}[i] + T_{t+1}[i] = X_t[i] + T_t[i]$. If there were no tokens at i, then in effect, there are still no tokens at i and the number of units $X[i]$ increases. Since the node was already saturated at t and $T_t[i] = 0$, the increase of $X[i]$ does not violate the second condition of the lemma.

2. Note that j has to be unsaturated, i.e. $X_t[j] > D_t[j]$ (otherwise it would not be chosen by our algorithm). Since a unit is exchanged for a token, $X_{t+1}[j] + T_{t+1}[j] = X_t[j] + T_t[j]$. By inductive hypothesis, the latter term is equal to $F_q[j]$.

■

Lemma 7.7. *The number of tokens produced within any phase P_q is at most $\delta(F_q, F'_q)$.* [1]

Proof. Since the number of tokens does not decrease within a phase it is sufficient to show that the total number of tokens at the end of phase P_q is at most $\delta(F_q, F'_q)$. Let t be the last time step of P_q, i.e. $X_t = F'_q$. By Lemma 7.6, for any i it holds that $T_t[i] = 0$ or $T_t[i] = F_q[i] - X_t[i] = F_q[i] - F'_q[i]$. Therefore, $|T_t| \leq \sum_{i:F_q[i]>F'_q[i]}(F_q[i] - F'_q[i]) = \delta(F_q, F'_q)$. ■

Bounds for one phase of PB. The following lemma shows an essential upper bound on the cost of the PB algorithm in one phase.

Lemma 7.8. *Let UNF_q be a binary indicator variable denoting whether phase P_q was unfair. For any phase P_q, it holds that $C_{\text{PB}}(P_q) \leq \min\{n-1, \kappa_q\} \cdot \delta(F_q, F'_q) + \kappa_q \cdot \text{UNF}_{q-1}$.*

Proof. If the previous phase was unfair, then the cost of the reorganization is at most κ_q. By our earlier observation, the number of produced tokens is at most $\delta(F_q, F'_q)$ and therefore it is sufficient to show that each token is moved at most $\min\{n-1, \kappa_q\}$ times.

[1] It can be proven that the number of tokens produced is equal to $\delta(F_q, F'_q)$.

Fix any time step t. First, note that if node i borrows a unit from node j, then $i \notin \mathcal{U}_t$ and $j \in \mathcal{U}_t$. Therefore a token is sent from the node outside $\mathcal{U}_t$ to the node inside $\mathcal{U}_t$. Let y denote the size of set $\mathcal{U}_t$ when the first borrowing of the phase takes place. We have $y \leq n-1$ and $y \leq \kappa_q$. Since, by Lemma 7.3, $\mathcal{U}_{t+1} \subseteq \mathcal{U}_t$, a token can be sent only y times. This concludes the proof. ■

Sequences of fair phases. When we combine Lemmas 7.4 and 7.8, we can relate the cost of the optimal algorithm in two consecutive fair phases P_{q-1} and P_q to the cost of the algorithm PB in P_q. It is straightforward that these bounds hold for all phases of input sequences of the k-resource problem. Therefore the following lemma follows.

Lemma 7.9. *Fix any phase-based algorithm A and any input sequence σ. Assume that for any two consecutive fair phases P_{q-1}, P_q, it holds that $C_A(P_q) \leq \gamma(\kappa_q) \cdot \delta(F_q, F'_q)$. Then A is $O(\gamma(k))$-competitive for the k-resource problem.*

Proof. Fix any input sequence σ and let $P_1, P_2, \ldots, P_p$ be its division into phases. Since we are dealing with the k-resource problem, all these phases are fair and $\kappa_q = k$ for all $1 \leq q \leq p$. By Lemma 7.4 and 7.5, we get that the inequality $C_A(P_q) \leq O(\gamma(k)) \cdot \left[C_{\mathrm{OPT}}(P_{q-1}) + C_{\mathrm{OPT}}(P_q)\right]$ holds for any phase P_q where $2 \leq q \leq p-1$. Therefore,

$$\begin{aligned} C_A(\sigma) &\leq C_A(P_1) + C_A(P_p) + \sum_{q=2}^{p-1} C_A(P_q) \\ &\leq 2 \cdot O(k \cdot \gamma(k)) + \sum_{q=2}^{p-1} O(\gamma(k)) \cdot C_{\mathrm{OPT}}(P_q) + \sum_{q=1}^{p-2} O(\gamma(k)) \cdot C_{\mathrm{OPT}}(P_q) \\ &\leq 2 \cdot O(\gamma(k)) \cdot C_{\mathrm{OPT}}(\sigma) + O(k \cdot \gamma(k)) \ . \end{aligned}$$

■

Theorem 7.10. *Any (deterministic) phase-based algorithm* PB *is $O(\min\{n,k\})$-competitive for the k-resource problem.*

7.2.2. Randomized phase-based algorithm for DR problem

In this section we show a randomized phase-based algorithm for the DR problem, which — on k-resource instances — is $O(\log(\min\{k,n\}))$-competitive. We only have to specify the $\mathcal{U}$-rule: the algorithm chooses a node with the smallest number of tokens. If there is more than one such node, it is chosen uniformly at random. We denote the resulting algorithm by PBR.

Distribution of tokens. Our goal is to define the whole game between the adversary and the algorithm in terms of changes in token positions. For each time step t we introduce the notion of set $\mathcal{U}'_t = \{i \in V : X_t[i] > D_t[i]\}$, i.e. the set of all nodes which are unsaturated *after* the algorithm moves its unit. Similarly to the Lemma 7.3, we get that $\mathcal{U}'_{t+1} \subseteq \mathcal{U}_{t+1} \subseteq \mathcal{U}'_t$. Let ℓ_t denote the minimum number of tokens at a node from $\mathcal{U}'_t$. We can show that each node from $\mathcal{U}'_t$ has either ℓ_t or $\ell_t + 1$ tokens.

Lemma 7.11. *In step t the number of tokens at any node is at most* $\ell_t + 1$.

Proof. We prove the lemma by induction on the number of time steps. At the very beginning of the phase, there are no tokens, $\ell_t = 0$ and the lemma trivially holds. Assume that the condition holds for time step t. Since in time step $t+1$ a new token can appear only in set $\mathcal{U}_{t+1} \subseteq \mathcal{U}'_t$ and it can only appear at the nodes with ℓ_t tokens, the new number of tokens at any node is at most $\ell_t + 1$. The number of tokens at set $\mathcal{U}'$ can only increase, and thus this amount is at most $\ell_{t+1} + 1$. ■

There are two reasons for a node $i \in \mathcal{U}'$ to become saturated and in effect excluded from $\mathcal{U}'$. The first one occurs when the demand $D[i]$ increases to the level equal to $X[i]$. The second one occurs when a unit from i is borrowed by some other node and in effect $X[i]$ decreases to the value of $D[i]$. Although similar, these cases substantially differ. In the former case, the choice of node is made by the adversary; in the latter case, the node is chosen in random fashion by the algorithm and not known precisely to the adversary.

The key to the analysis is the set $\mathcal{A}_t = \mathcal{U}'_t \cup \{i \in V : T_t[i] = \ell_t + 1\}$. Informally, $\mathcal{A}_t$ contains all the nodes which are either unsaturated or these for which the adversary cannot be sure that they are saturated. Obviously, each node from $\mathcal{A}_t$ contains ℓ_t or $\ell_t + 1$ tokens. The following lemma redefines the whole game between the algorithm and the adversary in terms of token shifting.

Lemma 7.12. *In step* $t + 1$, *there are two possible actions concerning tokens and set* $\mathcal{A}$ *that can be incurred by the adversary.*

A. A token is moved from $i \notin \mathcal{A}_t$ *to* $j \in \mathcal{A}_t$. *If i had no tokens, then one token was produced at i.*

B. A node i is removed from set $\mathcal{A}_t$. *If i had* $\ell_t + 1$ *tokens,* ℓ_t *tokens remain at i and one is moved to* $j \in \mathcal{A}_t$.

In both actions, node j is chosen uniformly at random amongst these nodes of $\mathcal{A}_t$ *which have* ℓ_t *tokens. Additionally, as a result of Action A, set* $\mathcal{A}$ *may decrease.*

Proof. When can the distribution of tokens or the set $\mathcal{A}$ change? Obviously, a necessary condition is that the vector D has to change (increase), because otherwise the algorithm makes no action at all. Let i be the node at which the demand D increased, i.e. $D_{t+1}[i] = D_t[i] + 1$. We consider several cases.

- $i \notin \mathcal{A}_t$. Since $X_t[i] = D_t[i]$, $X_t[i] < D_{t+1}[i]$, and node i has to borrow a unit. In this case a token is produced at i if necessary, and moved from node i to node j chosen randomly from $\mathcal{U}_{t+1}$. We show that in this case Action A takes place.

  Let us take a closer look at the structure of $\mathcal{U}_{t+1}$ set. We have $\mathcal{U}_{t+1} \subseteq \mathcal{U}'_t \subseteq \mathcal{A}_t$, and since at the beginning of step $t+1$ the demand increased outside $\mathcal{A}_t$, $\mathcal{U}_{t+1} = \mathcal{U}'_t$. By Lemma 7.11, $\mathcal{U}'_t$ consists of nodes with either ℓ_t or $\ell_t + 1$ tokens, and so does $\mathcal{U}_{t+1}$. Additionally, the $\mathcal{U}_{t+1}$ nodes which have ℓ_t tokens are exactly the same nodes as the ones from $\mathcal{A}_t$ with ℓ_t tokens. Therefore, j is in fact chosen uniformly at random amongst $\mathcal{A}_t$ nodes with ℓ_t tokens.

  If after moving a token to $\mathcal{A}_t$ in step $t + 1$, all nodes from $\mathcal{A}$ have $\ell_t + 1$ tokens, $\ell_{t+1} = \ell_t + 1$. In effect set $\mathcal{A}$ changes: all saturated nodes are removed from it.

- $i \in \mathcal{A}_t$. Let P_q be the current phase. By Lemma 7.6, we get that $X_t[i] + T_t[i] = F_q[i]$. We first show that

$$D_{t+1}[i] \leq F_q[i] - \ell_t \ . \tag{7.1}$$

  It is sufficient to show that $D_t[i] < F_q[i] - \ell_t$. If $T_t[i] = \ell_t + 1$, then $D_t[i] \leq X_t[i] = F_q[i] - \ell_t - 1$. Otherwise $T_t[i] = \ell_t$, and thus $i \in \mathcal{U}'_t$, which implies $D_t[i] < X_t[i]$, and in effect $D_t[i] < F_q[i] - \ell_t$.

  By (7.1), it suffices to consider two sub cases, depending on the demand at node i.

  - $D_{t+1}[i] \leq F_q[i] - \ell_t - 1$. Since $X_t[i] = F_q[i] - T_t[i] \geq F_q[i] - \ell_t - 1$, the algorithm has nothing to do. Although such a change may cause i to become saturated, this happens only if i has $\ell_t + 1$ tokens, and therefore set $\mathcal{A}$ does not change.
  - $D_{t+1}[i] = F_q[i] - \ell_t$. We show that this case corresponds to one Action B. If i has ℓ_t tokens, then since $X_t[i] = F_q[i] - \ell_t$, the algorithm does nothing. If i has $\ell_t + 1$ tokens, it has $F_q[i] - \ell_t - 1$ units and has to borrow one. In effect it chooses randomly a node j from the set $\mathcal{U}_{t+1}$ and a token is sent from i to j. It holds that $\mathcal{U}_{t+1} = \mathcal{U}'_t \setminus \{i\}$, and thus, as above, the algorithm chooses in fact node j uniformly at random amongst these nodes of $\mathcal{A}_t$ which have ℓ_t tokens. In both cases i becomes saturated and therefore removed from sets $\mathcal{U}'$ and $\mathcal{A}$.

■

Note that the adversary can compute the set $\mathcal{A}_t$, but since it is oblivious it does not know exactly which nodes have ℓ_t tokens and which $\ell_t + 1$. Let r_t be the number of $\mathcal{A}_t$ nodes with $\ell_t + 1$ tokens. In the appendix, we prove that the nodes with higher number of tokens are distributed uniformly in $\mathcal{A}_t$.

Lemma 7.13. *The probability that a node from $\mathcal{A}_t$ has $\ell_t + 1$ tokens is equal to $r_t/|\mathcal{A}_t|$.*

Proof. We prove a stronger condition. For a set $\mathcal{I} \subseteq \mathcal{A}_t$, such that $|\mathcal{I}| = r_t$, by $\mathcal{E}^t_{\mathcal{I}}$ we denote an event that at the end of round t, $\mathcal{I}$ is the set of nodes with $\ell_t + 1$ tokens. We show inductively that

$$\mathbf{Pr}\left[\mathcal{E}^t_{\mathcal{I}}\right] = \frac{1}{\binom{|\mathcal{A}_t|}{r_t}} \ . \tag{7.2}$$

Before we prove the equality above, we show how the lemma follows from it. Fix any node $x \in \mathcal{A}_t$. There are exactly $\binom{|\mathcal{A}_t|-1}{r_t-1}$ sets with r_t elements which contain x. Since $\mathcal{E}^t_{\mathcal{I}}$ are mutually exclusive events, the probability that that x contains $\ell_t + 1$ tokens is equal to $\binom{|\mathcal{A}_t|-1}{r_t-1} \cdot \frac{1}{\binom{|\mathcal{A}_t|}{r_t}} = \frac{r_t}{|\mathcal{A}_t|}$

At the beginning, there are no tokens in the graph, $r_t = 0$ and (7.2) trivially holds. Assume that in round t, (7.2) is fulfilled for all $\mathcal{I}$, such that $|\mathcal{I}| = r_t$.

If Action A occurs, then one token is added to a node $j \in \mathcal{A}_t$ with $T_t[j] = \ell_t$. If after this action all nodes have $\ell_t + 1$ tokens, $\ell_{t+1} = \ell_t + 1$, $r_{t+1} = 0$ and the condition (7.2) trivially holds. Otherwise $r_{t+1} = r_t + 1$. Choose any set $\mathcal{I}' \subseteq \mathcal{A}_{t+1} = \mathcal{A}_t$, such that $|\mathcal{I}'| = r_{t+1}$. Then

$$\begin{aligned}
\mathbf{Pr}\left[\mathcal{E}^{t+1}_{\mathcal{I}'}\right] &= \mathbf{Pr}\left[\exists_{x\in\mathcal{I}'} \ \mathcal{E}^t_{(\mathcal{I}'\setminus\{x\})} \ \wedge \ \text{token was added to } x \text{ in round } t+1\right] \\
&= r_{t+1} \cdot \left[\frac{1}{\binom{|\mathcal{A}_t|}{r_t}} \cdot \frac{1}{|\mathcal{A}_t| - r_t}\right] \\
&= \frac{1}{\binom{|\mathcal{A}_{t+1}|}{r_{t+1}}} \ .
\end{aligned}$$

If Action B occurs, then we can divide such action into two parts. In the first part, a node (either with ℓ_t or with $\ell_t + 1$ tokens) is removed from $\mathcal{A}_t$. In the second part a token is added to the reduced $\mathcal{A}$ set. The events of the latter type were already analyzed above and thus it remains to consider the removal of nodes from $\mathcal{A}_t$. Assume that the removed node i had ℓ_t tokens. Then $r_{t+1} = r_t$. Pick any set $\mathcal{I}' \subseteq \mathcal{A}_{t+1}$, such that $|\mathcal{I}'| = r_{t+1}$. It holds that

$$\mathbf{Pr}\left[\mathcal{E}^{t+1}_{\mathcal{I}'}\right] = \mathbf{Pr}\left[\mathcal{E}^t_{\mathcal{I}'} \mid i \text{ has } \ell_t \text{ tokens}\right] = \frac{\mathbf{Pr}\left[\mathcal{E}^t_{\mathcal{I}'} \ \wedge \ i \text{ has } \ell_t \text{ tokens}\right]}{\mathbf{Pr}\left[i \text{ has } \ell_t \text{ tokens}\right]} \ ,$$

and since event $\mathcal{E}^t_{\mathcal{I}'}$ implies that i has ℓ_t tokens,

$$\mathbf{Pr}\left[\mathcal{E}^{t+1}_{\mathcal{I}'}\right] = \frac{\mathbf{Pr}\left[\mathcal{E}^t_{\mathcal{I}'}\right]}{\mathbf{Pr}\left[i \text{ has } \ell_t \text{ tokens}\right]} = \frac{1/\binom{|\mathcal{A}_t|}{r_t}}{\frac{|A_t|-r_t}{|A_t|}} = \frac{1}{\binom{|\mathcal{A}_t|-1}{r_t}} = \frac{1}{\binom{|\mathcal{A}_{t+1}|}{r_{t+1}}} \ .$$

Otherwise, the removed node i had ℓ_{t+1} tokens. In this case we pick any set $\mathcal{I}' \subseteq \mathcal{A}_{t+1}$, such that $|\mathcal{I}'| = r_{t+1} = r_t - 1$. The proof is analogous, i.e. it holds that

$$\mathbf{Pr}\left[\mathcal{E}^{t+1}_{\mathcal{I}'}\right] = \mathbf{Pr}\left[\mathcal{E}^t_{(\mathcal{I}'\cup\{i\})} \mid i \text{ has } \ell_t + 1 \text{ tokens}\right] = \frac{\mathbf{Pr}\left[\mathcal{E}^t_{(\mathcal{I}'\cup\{i\})} \ \wedge \ i \text{ has } \ell_t + 1 \text{ tokens}\right]}{\mathbf{Pr}\left[i \text{ has } \ell_t + 1 \text{ tokens}\right]} \ ,$$

and since event $\mathcal{E}^t_{(\mathcal{I}'\cup\{i\})}$ implies that i has $\ell_t + 1$ tokens,

$$\mathbf{Pr}\left[\mathcal{E}^{t+1}_{\mathcal{I}'}\right] = \frac{\mathbf{Pr}\left[\mathcal{E}^t_{(\mathcal{I}'\cup\{i\})}\right]}{\mathbf{Pr}\left[i \text{ has } \ell_t + 1 \text{ tokens}\right]} = \frac{1/\binom{|\mathcal{A}_t|}{r_t}}{\frac{r_t}{|\mathcal{A}_t|}} = \frac{1}{\binom{|\mathcal{A}_t|-1}{r_t-1}} = \frac{1}{\binom{|\mathcal{A}_{t+1}|}{r_{t+1}}} \ .$$

Thus, for either action (7.2) holds for step $t+1$. ■

Number of token movements. In order to bound the expected number of token movements in one phase, we construct a potential function Φ, which relates this amount to the current distribution of tokens.

We fix any phase P_q and any time step $t \in P_q$. Let $L_t = \sum_{i \notin \mathcal{A}_t} T_t[i]$, the number of tokens outside set $\mathcal{A}_t$. Let

$$\Phi_t = |T_t| - \delta(F_q, F'_q) \cdot H_{|\mathcal{A}_t|} - L_t \ ,$$

where $H_i = \sum_{j=1}^{i} 1/j$. Below we show that the expected cost of PBR in one step is bounded by the change in the potential.

Lemma 7.14. *In time step $t+1$, $\mathbf{E}[C_{\mathrm{PBR}}(t+1)] \leq \mathbf{E}[\Phi_{t+1} - \Phi_t]$.*

Proof. By Lemma 7.12, we have to consider two types of events.

Assume that Action A occurs. If a token is produced at node $i \notin \mathcal{A}_t$, both $|T|$ and L increase by 1, which does not change the potential. The cost of moving a token is paid by the decrease of L by 1. Finally, set $\mathcal{A}$ may decrease, but this can only increase the change in the potential.

Assume that Action B occurs. Then $\mathcal{A}_{t+1} = \mathcal{A}_t \setminus \{i\}$ and ℓ_t tokens are removed from set $\mathcal{A}$. $\mathbf{E}[C_{\mathrm{PBR}}(t+1)] = r_t/|\mathcal{A}_t|$ because $r_t/|\mathcal{A}_t|$ is the probability of moving a token. Therefore, the expected change in the potential is equal to

$$\mathbf{E}[\Phi_{t+1} - \Phi_t] = \delta(F_q, F'_q) \cdot [H_{|\mathcal{A}_t|} - H_{|\mathcal{A}_{t+1}|}] - \ell_t = \frac{\delta(F_q, F'_q)}{|\mathcal{A}_t|} - \ell_t \leq \frac{r_t}{|\mathcal{A}_t|},$$

where the last inequality holds because the number of tokens in $\mathcal{A}_t$, $\ell_t \cdot |\mathcal{A}_t| + r_t$, is at most $\delta(F_q, F'_q)$. ■

Since it always holds that $|\mathcal{A}_t| \leq \min\{n, \kappa_q\}$ and $|T_t| \leq \delta(F_q, F'_q)$, the absolute value of Φ is bounded by $O(\log(\min\{n, \kappa_q\})) \cdot \delta(F_q, F'_q)$. By Lemma 7.14, the same bound holds for the expected number of token movements in phase P_q, and thus we get the following lemma.

Lemma 7.15. *For any phase* P_q, $\mathbf{E}[C_{\mathrm{PBR}}(P_q)] = O(\log(\min\{n, \kappa_q\})) \cdot \delta(F_q, F'_q) + \kappa_q \cdot \mathrm{UNF}_{q-1}$.

We note that the Lemma 7.9 holds analogously also for randomized algorithms. Thus, we get the following conclusion.

Theorem 7.16. PBR *is* $O(\log(\min\{n, k\}))$*-competitive for the* k*-resource problem.*

7.3. Uniformly decomposable metric spaces

Up to this point we have been considering the k-resource problem for uniform metric spaces. We now turn our attention to a metric space $(\mathcal{X}, d)$ which is uniformly decomposable into a set Λ of components, each being a uniform metric space. The distances between components are denoted by Γ and $z = |\Lambda|$.

Outline of the solution. To solve the k-resource problem on a uniformly decomposable metric space, we have to care about an appropriate resource assignment on two levels: the available resources must be split accordingly between the components of the metric space and they must be assigned to single points of the metric space within each of the components.

The resource assignment in each of the components is managed by separate instances of algorithms for the dynamic-resource problem (we use the algorithm PBR introduced in the previous section). Note that if we look only at the higher level and we treat each component as one point of the space, we have a usual k-resource game on a uniform metric. However, if we solve it naively, neglecting what really happens inside of particular components, we obtain an inefficient algorithm. For example, if the whole activity takes place in one component, but the total demand in this component is constant, the naive algorithm on higher level does nothing and may assign much fewer resources to this component than the optimal algorithm. Therefore, the algorithm PBR working inside this particular component has no chance to be competitive.

Thus, the PBR algorithms running inside the components have to give the higher level assignment algorithm *hints* in which components the resource demand changes in a significant manner. Nevertheless, even with these hints, the problem seems to be difficult and therefore we give our algorithm a little bit more resources than to the optimal one (i.e. we use resource augmentation). This guarantees that there will be many phases in which PBR algorithms have at least as many resources as Opt. By the previous section, we know that the ratio between their costs in such phases is at most logarithmic in k and in the number of nodes in the specific component.

Upper level of DMR. We introduce an algorithm DMR. It works by using separate instances PBR_λ of the algorithm PBR in each component $\lambda \in \Lambda$. After the adversary changes the demands in component λ, the algorithm DMR checks whether more resources must be moved to component λ. If necessary, this movement is performed and afterward algorithm PBR_λ is used to handle that request with the resources available locally in λ.

Let $F_q(\lambda)$, $D_t(\lambda)$, $I_q(\lambda)$, and $k_t(\lambda)$ denote, respectively, the variables F_q, D_t, I_q, and k_t for the algorithm PBR_λ (as introduced in the analysis of PBR). For any step t belonging to phase q we define a *padding* $g_t(\lambda)$ as

$$g_t(\lambda) = \sum_{i:D_t(\lambda)[i]>F_q(\lambda)[i]} (D_t(\lambda)[i] - F_q(\lambda)[i]) \ . \tag{7.3}$$

Let $w[\lambda]$ denote the total demand for resources in a component λ, i.e. $w_t[\lambda] = \sum_{p\in\lambda} d_t[p]$. The assignment of resources to components is performed by an algorithm which depends on changes of the padding g. In particular, we construct the following sequence of vectors:

$$W_t[\lambda] = \begin{cases} \max\{W_{t-1}[\lambda] + 1/\Gamma, w_t[\lambda]\} & \text{if } t > 2 \text{ and } g_{t-1}(\lambda) > g_{t-2}(\lambda), \\ \max\{W_{t-1}[\lambda], w_t[\lambda]\} & \text{otherwise.} \end{cases}$$

This recursive definition is still incomplete, since we have to give some start conditions. Note that by this definition only two elements of W can increase per time step. Using the values of W the algorithm DMR divides the input sequence σ into epochs $E_1, E_2, \ldots$ The first epoch starts with the first time step. At the beginning of each epoch we set $W = w$. An epoch ends at the end of a step t (after PBR executed all its actions in t) in which $|W_t| \geq (1+c)\cdot k - 2$. Since $|W|$ can increase by at most $1 + 1/\Gamma$ per step, at the very end of the epoch $|W| \leq (1+c)\cdot k$ when the epoch ends in t. For convenience we call each increase of W due to the condition $g_{t-1}(\lambda) > g_{t-2}(\lambda)$ a *hit* in component λ.

Let a vector B_t denote the number of resources available in the components. Obviously $|B_t| = (1+c)\cdot k$. In step t we *assign* exactly $\lfloor W_t[\lambda]\rfloor$ resources to component λ by setting $k_t(\lambda) = \lfloor W_t[\lambda]\rfloor$. If there are too few resources in λ (i.e. $\lfloor W_t[\lambda]\rfloor > B_{t-1}[\lambda]$), then a resource item is moved from another component. This is always possible since we have $(1+c)\cdot k$ resource units available and $\sum_{\lambda\in\Lambda}\lfloor W_t[\lambda]\rfloor \le (1+c)\cdot k$ by the way epochs are constructed.

In effect, DMR constructs a part of the input sequence for the dynamic-resource problem solved by PBR_λ by changing the number of assigned resources. In any time step, the DMR algorithm moves resources between components prior to the algorithms PBR_λ. This assures that each PBR_λ has $\lfloor W_t[\lambda]\rfloor \ge w_t[\lambda]$ resources available to fulfill the demand. The PBR_λ algorithm may only use the assigned resources to fulfill the demand, and not all the resources really available in the component. Such construction implies that not all resources are always used; this appears counterproductive at first glance, but it keeps the division into epochs deterministic and dependent only on the input sequence.

Fair timespans. We introduce a notion of fair timespans, which is crucial to the analysis of the DMR algorithm. By $\widetilde{W}_t[\lambda]$ we denote the number of resources maintained by OPT in component λ. A timespan $\bar{t}$ of an input sequence is defined as a contiguous sequence of time steps. We call a timespan $\bar{t}$, starting at time step s, *fair for component* λ if $\lfloor W_s[\lambda]\rfloor \ge \widetilde{W}_s[\lambda]$ and for each step $t \in \bar{t}$, $\lfloor W_t[\lambda]\rfloor \ge \lfloor W_s[\lambda]\rfloor$ and $\widetilde{W}_t[\lambda] \le \widetilde{W}_s[\lambda]$.

Analysis of DMR. Before we analyze the cost of DMR and OPT inside of epochs, we want to establish the fact, that the division into epochs depends only on σ. Epochs are determined by the values of W, and it is easily seen that the values of W_t depend only on W_{t-1} and g_{t-1}. By an easy induction one can see that W_t does not depend on the random choices of the algorithms PBR_λ, and therefore the division into epochs is deterministic. The following lemma shows how the values g_t depend on σ.

Lemma 7.17. *In the DR problem the value g_t depends only on the input sequence σ (including the values of k) up to step t and the initial resource vector X_0. In particular, it does not depend on the $\mathcal{U}$-rule used.*

Proof. The division into phases does not depend on the algorithm and nor does D_t. Thus, it is sufficient to show that the values F_q also depend only on σ. We prove this inductively on the number of phases. At the beginning, $F_1 = X_0$. Assume that the condition holds for the first q phases. If phase P_q is fair, then at its end $X = D$ and then surplus nodes at v_S are (possibly) added to X so that $|X| = \kappa_{q+1}$. Thus, F_q does not

depend on PB choices. On the other hand, if P_q is unfair, then during the reorganization the resource vector is created solely on the basis of the demand d_s and k_s, where s is the first step of phase P_{q+1}. ■

Corollary 7.18. *The division into epochs is deterministic, i.e. it depends only on σ.*

Using Lemma 7.15 and observing that with each hit $|W|$ increases by $1/\Gamma$, we may bound the cost of DMR in each epoch. A lower bound on the cost of Opt relies on the notion of *fair timespans*. In such a timespan, the algorithm PBR_λ has at least the same number of resources as Opt, and therefore the phases in this timespan are fair. Since DMR has $(1+c)\cdot k$ resources available, we can guarantee that there exist fair timespans of appropriate length during an epoch. In a fair timespan, the PBR_λ algorithm is competitive against Opt, and thus each hit implies that Opt has incurred some cost.

We denote all costs by DMR and Opt due to transferring resources between components by $C^{\mathrm{T}}_{\mathrm{DMR}}$ and $C^{\mathrm{T}}_{\mathrm{OPT}}$, respectively. Then it holds that $C_{\mathrm{DMR}}(\sigma) = C^{\mathrm{T}}_{\mathrm{DMR}}(\sigma) + \sum_{\lambda\in\Lambda} C_{\mathrm{PBR}_\lambda}(\sigma|_\lambda)$ and $C_{\mathrm{OPT}}(\sigma) = C^{\mathrm{T}}_{\mathrm{OPT}}(\sigma) + \sum_{\lambda\in\Lambda} C_{\mathrm{OPT}_\lambda}(\sigma|_\lambda)$, where $\sigma|_\lambda$ denotes the part of input relevant for component λ and with the number of assigned resources set by the algorithm DMR.

The intuitions above are formalized in the two following lemmas.

Lemma 7.19. *Let $z \le c\cdot\Gamma$ and $\Gamma \ge k$. Then for any epoch E_y,*

$$\mathbf{E}[C_{\mathrm{DMR}}(E_y)] \le (1+c)\cdot k\cdot\Gamma\cdot O(\log(\min\{n,k\}))\ ,$$

where the expectation is taken w.r.t. all random choices made by the algorithms PBR_λ.

Proof. The number of resource transfers between components within epoch E_y is determined by the changes in $|W|$. The algorithm DMR moves a resource between components only if $\lfloor W\rfloor$ has increased and this can happen at most $(1+c)\cdot k$ times during an epoch. Thus,

$$C^{\mathrm{T}}_{\mathrm{DMR}}(E_y) \le (1+c)\cdot k\cdot\Gamma\ . \tag{7.4}$$

Note that $|W|$ grows by at least $1/\Gamma$, implying that the total number of hits issued in a phase is at most $(1+c)\cdot k\cdot\Gamma$. Consider now a specific component λ and let $h_\lambda(E_y)$ denote the number of hits issued in component λ within epoch E_y. Denote by $P_p,\ldots,P_{p'}$ the smallest set of phases by PBR_λ which contain the epoch E_y. Then it holds that

$$\sum_{q=p+1}^{p'-1} \delta(F_q(\lambda), F'_q(\lambda)) \le h_\lambda(E_y)+1\ ,$$

where the additional cost of 1 comes from the fact that an increase of g in the last time step of E_y may contribute to a hit in E_{y+1} instead of E_y. Let $\mathcal{I}_\lambda = \sum_{q=p}^{p'} \text{UNF}_{q-1}(\lambda)$ be the number of unfair phases in $P_{p-1}, \ldots, P_{p'-1}$. The expected cost of PBR_λ in epoch E_y is, by Lemma 7.15, bounded by

$$\begin{aligned}\mathbf{E}[C_{\text{PBR}_\lambda}(E_y|_\lambda)] &\leq \sum_{q=p}^{p'} O(\log(\min\{n,k\})) \cdot \delta(F_q(\lambda), F'_q(\lambda)) + k \cdot \text{UNF}_{q-1}(\lambda) \\ &\leq O(\log(\min\{n,k\})) \cdot \left(2 \cdot k + h_\lambda(E_y) + 1\right) + k \cdot \mathcal{I}_\lambda \ ,\end{aligned}$$

since $\delta(F_q(\lambda), F'_q(\lambda)) \leq k$ for any phase q. Note that a phase in component λ is unfair only if at least one resource is removed from λ during that phase; we map each unfair phase to such a transfer. All the phases $P_{p+1}, \ldots, P_{p'-1}$ are mapped to a transfer inside E_y; P_{p-1} and P_p are possibly mapped to a transfer outside E_y. Unfair phases from different components are mapped to different transfers, and therefore if we sum these phases over all the components, we get $\sum_{\lambda\in\Lambda} \mathcal{I}_\lambda \leq (1+c) \cdot k + 2z$. By summing up over all components we obtain

$$\begin{aligned}\sum_{\lambda\in\Lambda} \mathbf{E}[C_{\text{PBR}_\lambda}(E_y|_\lambda)] &\leq O(\log(\min\{n,k\})) \cdot \left(2 \cdot k \cdot z + \sum_{\lambda\in\Lambda} h_\lambda(E_y) + z\right) + k \cdot \sum_{\lambda\in\Lambda} \mathcal{I}_\lambda \\ &\leq O(\log(\min\{n,k\})) \cdot ((4 \cdot k + 1) \cdot z + 2(1+c) \cdot k \cdot \Gamma) \ , \qquad (7.5)\end{aligned}$$

since $\sum_{\lambda\in\Lambda} h_\lambda(E_y) \leq (1+c) \cdot k \cdot \Gamma$ and $\Gamma \geq k$. Combining Eq. (7.4) and (7.5), we obtain

$$\mathbf{E}[C_{\text{DMR}}(E_y)] \leq (1+c) \cdot k \cdot \Gamma \cdot O(\log(\min\{n,k\})) \ ,$$

since by assumption $z \leq c \cdot \Gamma$. ■

Lemma 7.20. *Let $z \leq c \cdot \Gamma/4$. Then for any epoch E_y, $C_{\text{OPT}}(E_y) = \Omega(c \cdot k \cdot \Gamma)$.*

Proof. Let e be the last time step of epoch E_y. We define $\eta[\lambda] = W_e[\lambda]$, $\tilde{\eta}[\lambda] = \max_{t\in E_y} \widetilde{W}_t[\lambda]$, and $\tilde{k} = |\tilde{\eta}|$. Note that $\sum_{\lambda\in\Lambda} \eta[\lambda] \geq (1+c) \cdot k - 2$. Since OPT has only k balls available, it has moved at least $\tilde{k} - k$ balls during a phase, and therefore

$$C^{\text{T}}_{\text{OPT}}(\sigma) \geq (\tilde{k} - k) \cdot \Gamma \ . \qquad (7.6)$$

We now want to lower bound the cost of OPT_λ. Let λ be a component with $\eta[\lambda] > \tilde{\eta}[\lambda]$. Let s be the time step when $\lfloor W_s[\lambda] \rfloor \geq \tilde{\eta}[\lambda]$ for the first time in epoch E_y. Since $\widetilde{W}_t[\lambda] \leq \tilde{\eta}[\lambda]$ for all $t \in E_y$, the timespan $[s, e]$ is fair for λ; we denote it by $\bar{t}_\lambda$.

Since Opt is able to serve the demands, $w_t[\lambda] \leq \tilde{\eta}[\lambda]$, and therefore the increase of $W[\lambda]$ from $\tilde{\eta}[\lambda]$ to $\eta[\lambda]$ could have only happened due to hits to λ. Thus, $h_\lambda(\bar{t}) \geq (\eta[\lambda] - \tilde{\eta}[\lambda]) \cdot \Gamma$, where by $h_\lambda(\bar{s})$ we denote the number of hits in component λ during timespan $\bar{s}$.

Let $P_p, \ldots, P_{p'}$ be the smallest set of phases by PBR_λ which contain $\bar{t}_\lambda$. During any finished phase P_q, the value of $g_t(\lambda)$ increases by 1 from 0 to $\delta(F_q(\lambda), F'_q(\lambda))$, and each such increase results in a hit. Note that an increase of $g(\lambda)$ results in a hit one time step later, so that an increase in the last step of E_{y-1} contributes to a hit in the epoch E_y. Thus, $h_\lambda(P_q) = \delta(F_q(\lambda), F'_q(\lambda))$ for all phases and therefore $h_\lambda(\bar{t}_\lambda) \leq 1 + \sum_{q=p}^{p'} \delta(F_q(\lambda), F'_q(\lambda))$. Since $\delta(F_q(\lambda), F'_q(\lambda)) \leq k$ for any q, it holds that

$$\sum_{q=p+2}^{p'-1} \delta(F_q(\lambda), F'_q(\lambda)) \geq h_\lambda(\bar{t}_\lambda) - 3 \cdot k - 1 \,.$$

Since $\bar{t}_\lambda$ is a fair timespan, all phases completely contained in $\bar{t}_\lambda$ are fair. That is, P_q for $q \in \{p+1, \ldots, p'-1\}$ are fair. Then, by Lemma 7.4 and 7.5, it holds that

$$C_{\text{OPT}_\lambda}(E_y) \geq \sum_{q=p+2}^{p'-1} \delta(\tau_{q-1}, \tau_q) \geq \sum_{q=p+2}^{p'-1} \tfrac{1}{2} \cdot \delta(F_q(\lambda), F'_q(\lambda)) \geq \tfrac{1}{2} \cdot (h_\lambda(\bar{t}) - 3 \cdot k - 1) \,.$$

Summing up over all $\lambda \in \Lambda$, we obtain

$$\begin{aligned} \sum_{\lambda \in \Lambda} 2C_{\text{OPT}_\lambda}(E_y) &\geq \sum_{\lambda : \eta[\lambda] > \tilde{\eta}[\lambda]} (h_\lambda(\bar{t}_\lambda) - 3k - 1) \geq \left(\sum_{\lambda : \eta[\lambda] > \tilde{\eta}[\lambda]} (\eta[\lambda] - \tilde{\eta}[\lambda]) \cdot \Gamma \right) - (3 \cdot k + 1) \cdot z \\ &\geq \left(\sum_{\lambda \in \Lambda} \eta[\lambda] - \sum_{\lambda \in \Lambda} \tilde{\eta}[\lambda] \right) \cdot \Gamma - (3 \cdot k + 1) \cdot z \\ &\geq ((1+c) \cdot k - 2 - \tilde{k}) \cdot \Gamma - (3 \cdot k + 1) \cdot z \quad . \end{aligned} \tag{7.7}$$

Combining (7.6) and (7.7), and by the fact $z \leq c \cdot \Gamma/4$, we can show the statement of the lemma. ■

The following theorem follows easily since each input sequence is deterministically partitioned into epochs and DMR is competitive for every finished epoch by the two former lemmas. The last potentially unfinished epoch contributes only an additive term.

Theorem 7.21. *Let $(\mathcal{X}, \xi)$ be a uniformly decomposable metric space with component set Λ. Then* DMR *is $O((1 + 1/c) \cdot \log(\min\{n, k\}))$-competitive for the k-resource problem on $(\mathcal{X}, \xi)$ with c-resource augmentation, $|\Lambda| \leq c \cdot \Gamma/4$ and $\Gamma \geq k$.*

Corollary 7.22. *Theorem 7.21 holds analogously for the k-server problem by the reduction from Lemma 7.1.*

CHAPTER 8

Summary and Outlook

In this thesis we have lied the algorithmic foundations for establishing a service which assures connectivity in a sparse network. Surprisingly, although research on wireless network is a very common topic, the most often found assumption is that networks are connected. Though, everyday experience with networks commonly used is that they are rather prone to disconnections. This makes it important to design schemes which can handle these disconnections, either by allowing the network to gracefully degrade performance, or as presented in this thesis to prevent disconnections. The approach of using mobile network participants for active maintenance of a connected network graph seems promising, although many tasks still remain challenging. We will briefly go through the three main topics handled in this thesis and outline their importance and deficiencies, allowing for future work.

Organizing a communication chain. The question on how to organize a communication chain with simple, local strategies is not only important due to its application for the connectivity management scenario, central to this thesis. Rather, what we find to be very interesting, is its contribution to the research on swarm robotics. With the *Hopper* strategy we have been able to show that simple strategies for this problem are able to achieve an optimal performance (up to constant factors). This shows that local and distributed methods can provide very good results while contributing to simple and maintainable systems. An important insight into the future design of similar strategies is that the model of execution in sequential runs, employed in the *Hopper* strategies, brings visible advantages in comparison to a truly parallel execution as in *Go-To-The-Middle*. Furthermore, the *Hopper* strategies have shown that their non-converging instable behavior is helping to remove unnecessary relays from the chain, which has shown to be very hard in strictly converging strategies, like *Go-To-The-Middle*. We even conjecture that a lower bound of at least $\omega(n)$ in the static scenario can be shown for parallel and

converging strategies.

Our work on strategies maintaining communication chains is restricted to a scenario with one explorer and one base camp. Further research on this topic might encompass the development strategies, which are able to deal with several explorers, thus maintaining a Steiner tree instead of a chain. This implies that more than one explorer are sharing chain parts which connect them to the base camp. This already poses a problem, as this shared chain part must be able to supply new relays at a large speed to extend multiple chains, when the corresponding explorers all move simultaneously. We have been also restricting the terrain model by not allowing any obstacles to be placed on the plane. In an extended scenario, with obstacles on the plane, local strategies are hardly able to maintain a chain approximating the shortest distance between the explorer and base camp, as the topology of such a chain with respect to the obstacles may change very often. We envision the local strategies to be able to maintain a chain which is topologically equivalent to the movement path of the explorer with a performance similar to that of the obstacle-free case. Some preliminary results on this topic have been already shown in [DKMS07].

Minimizing the number of changes in support graph. In order to minimize the energy usage of relays required for changing the support graph, we have been investigating the ODMST problem. Apart from our scenario, the ODMST problem can be of interest for any setting, where minimum spanning tree are used as a data structure and any changes to this structure are costly. This occurs e.g. in networks where trees are used as a routing or overlay structure. Changing such a MST means that routing or configuration tables have to be broadcasted along the network and updated. Minimum spanning trees have been used in such scenarios for a long time, some recent examples may be found in [WCLF02, YCM+04]. It remains an open question whether the RANDMST algorithm can be adapted, using the marking mechanism found in MSTMARK, to work in the full ODMST setting, with increases and decreases of edge weights. We have found the adversarial model in ODMST problem to be too powerful, leading to high competitive ratios for online algorithms. Therefore, we look forward for results obtained by carefully restricting the possibilities of the adversary. A promising approach lies in restricting the graph to fulfill the triangle inequality or to be a subset of a grid. This could allow for an improvement of the competitive ratios of online algorithms especially in the approximated ODMST problem, while still being applicable for the connectivity problem considered in this thesis.

Managing spare relays. The k-resource problem is not only of interest for the management of relays. As we have already outlined, it is applicable for any resource management problem, where moving resources has some cost associated with it. Furthermore,

the solutions on uniformly decomposable metrics contribute to the set of metrics for which efficient results for the *k*-server problem are known. It remains an open problem whether the results shown can be applied to less restrictive decompositions. A natural extension would be to consider recursively and uniformly decomposable metrics, which are not restricted to only two layers of decomposition as shown here. The concept of resource augmention has shown to be especially useful for the case of decomposable metrics to provide a significant advantage for the online algorithm over the adversary. We believe this could be used to improve other results on decomposable metric spaces, like those shown in [Sei01]. In our results, we have been investigating the total energy expenditure of the relays during the execution of the input sequence. While this is an important factor, we are also interested in the time necessary to process an input sequence by our online algorithms. In order to measure time rather than energy, we define the cost of an algorithm in a time step to be themaximum over the distances by which resources aremoved in a time step. This implies, that the algorithm may move many resources “for free” during a time step, paying only for the movement on the largest distance. Modeling the cost in this way introduces significant changes for the behavior of the algorithm. Up to our knowledge, there exists no similar online problem which has been considered under this cost measure.

Bibliography

[AEGH98] P. K. Agarwal, D. Eppstein, L. J. Guibas, and M. Rauch Henzinger. Parametric and kinetic minimum spanning trees. In *Proc. of the 39th IEEE Symp. on Foundations of Computer Science (FOCS)*, page 596, Washington, DC, USA, 1998. IEEE Computer Society.

[AOSY99] H. Ando, Y. Oasa, I. Suzuki, and M. Yamashita. Distributed memoryless point convergence algorithm for mobile robots with limited visibility. In *IEEE Transactions on Robotics and Automation*, volume 15, pages 818–828, 1999.

[AP04] N. Agmon and D. Peleg. Fault-tolerant gathering algorithms for autonomous mobile robots. In *Proc. of the 15th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 1063–1071, 2004.

[Art64] E. Artin. *The Gamma Function*. Holt, Rinehart and Winston, 1964.

[BBM01] Y. Bartal, B. Bollobás, and M. Mendel. A Ramsey-type theorem for metric spaces and its applications for metrical task systems and related problems. In *Proc. of the 42nd IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 396–405, 2001.

[Bie05] M. Bienkowski. Dynamic page migration with stochastic requests. In *Proc. of the 17th ACM Symp. on Parallelism in Algorithms and Architectures (SPAA)*, pages 270–278, 2005.

[BK07] M. Bienkowski and J. Kutylowski. The *k*-resource problem on uniform and on uniformly decomposable metric spaces. In *Proc. of the 10th Int. Workshop on Algorithms and Data Structures (WADS)*, volume 4619 of *Lecture Notes in Computer Science*, pages 337–348, 2007.

[BKR+01] A. Borodin, J. Kleinberg, P. Raghavan, M. Sudan, and D. P. Williamson. Adversarial queuing theory. *Journal of the ACM*, 48(1):13–38, 2001.

[BLMN03] Y. Bartal, N. Linial, M. Mendel, and A. Naor. On metric Ramsey-type phenomena. In *Proc. of the 35th ACM Symp. on Theory of Computing (STOC)*, pages 463–472, 2003.

[BM05] Y. Bartal and M. Mendel. Randomized *k*-server algorithms for growth-rate bounded graphs. *Journal of Algorithms*, 55(2):192–202, 2005.

[BS89] D. L. Black and D. D. Sleator. Competitive algorithms for replication and migration problems. Technical Report CMU-CS-89-201, Carnegie Mellon University, Computer Science, 1989.

[CCJ90] B. N. Clark, C. J. Colbourn, and D. S. Johnson. Unit disk graphs. In *Discrete Mathematics*, volume 86, pages 165–177, 1990.

[CFPS03] M. Cieliebak, P. Flocchini, G. Prencipe, and N. Santoro. Solving the robots gathering problem. In *Proc. of the 30th Int. Colloq. on Automata, Languages and Programming (ICALP)*, pages 1181–1196, 2003.

[CH78] F. Chin and D. Houck. Algorithms for updating minimal spanning trees. *Journal of Computer and System Sciences*, 16:333–344, 1978.

[Cha00] B. Chazelle. A minimum spanning tree algorithm with inverse-ackermann type complexity. *Journal of the ACM*, 47(6):1028–1047, 2000.

[Cin75] E. Cinlar. *Introduction to Stochastic Processes*. Springer-Verlag, New York, 1975.

[CL06] B. Csaba and S. Lodha. A randomized on-line algorithm for the *k*-server problem on a line. *Random Structures and Algorithms*, 29(1):82–104, 2006.

[CLLR97] M. Chrobak, L. L. Larmore, C. Lund, and N. Reingold. A better lower bound on the competitive ratio of the randomized 2-server problem. *Information Processing Letters*, 63(2):79–83, 1997.

[CMN04] I. Chatzigiannakis, M. Markou, and S. Nikoletseas. Distributed circle formation for anonymous oblivious robots. In *Proc. of the 3rd Workshop on Efficient and Experimental Algorithms (WEA)*, volume 3059 of *Lecture Notes in Computer Science*, pages 159–174, 2004.

[CP02] M. Cieliebak and G. Prencipe. Gathering autonomous mobile robots. In *Proc. of the 9th Int. Colloq. on Structural Information and Communication Complexity (SIROCCO)*, volume 13 of *Proceedings in Informatics*, pages 57–72. Carleton Scientific, 2002.

[CP04a] R. Cohen and D. Peleg. Convergence of autonomous mobile robots with inaccurate sensors and movements. Technical Report MCS04-08, The Weizmann Institute of Science, 2004.

[CP04b] R. Cohen and D. Peleg. Robot convergence via center-of-gravity algorithms. In *Proc. of the 11th Int. Colloq. on Structural Information and Communication Complexity (SIROCCO)*, volume 3104 of *Lecture Notes in Computer Science*, pages 79–88, 2004.

[CP05] R. Cohen and D. Peleg. Convergence properties of the gravitational algorithm in asynchronous robot systems. *SIAM Journal on Computing*, 34(6):1516–1528, 2005.

[CT76] D. Cherito and R. E. Tarjan. Finding minimum spanning trees. *SIAM Journal on Computing*, 5(4), 1976.

[DK02] X. Défago and A. Konagaya. Circle formation for oblivious anonymous mobile robots with no common sense of orientation. In *Proc. of the 2nd ACM International Workshop on Principles of Mobile Computing (POMC)*, pages 97–104. ACM Press, 2002.

[DKK07] M. Dynia, M. Korzeniowski, and J. Kutylowski. Competitive maintenance of minimum spanning tree in dynamic graphs. In *Proc. of the 33rd International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)*, volume 4362 of *Lecture Notes in Computer Science*, pages 260–271. Springer-Verlag Berlin, 2007.

[DKLM06] M. Dynia, J. Kutylowski, P. Lorek, and F. Meyer auf der Heide. Maintaining communication between an explorer and a base station. In *Proc. of the 1st IFIP Int. Conf. on Biologically Inspired Cooperative Computing (BICC)*, IFIP, pages 137–146. Springer-Verlag Berlin, 2006.

[DKMS07] M. Dynia, J. Kutylowski, F. Meyer auf der Heide, and J. Schrieb. Local strategies for maintaining a chain of relay stations between an explorer and a base station. In *Proc. of the 19th ACM Symp. on Parallelism in Algorithms and Architectures (SPAA)*, pages 260–269. ACM Press, 2007.

[EGIN97] D. Eppstein, Z. Galil, G. F. Italiano, and A. Nissenzweig. Sparsification a technique for speeding up dynamic graph algorithms. *Journal of the ACM*, 44(5):669–696, 1997.

[Epp96] D. Eppstein. Spanning trees and spanners. Technical Report ICS-TR-96-16, University of California, 1996.

[FKL+91] A. Fiat, R. M. Karp, M. Luby, L. A. McGeoch, D. D. Sleator, and N. E. Young. Competitive paging algorithms. *Journal of Algorithms*, 12(4):685–699, 1991.

[FKN06] S. P. Fekete, R. Klein, and A. Nüchter. Online searching with an autonomous robot. *Computational Geometry: Theory and Applications*, 34:102–115, 2006.

[FPSW99] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Hard tasks for weak robots: The role of common knowledge in pattern formation by autonomous mobile robots. In *Proc. of the 10th Int. Symp. on Algorithms and Computation (ISAAC)*, pages 93–102, 1999.

[Fre83] G. N. Frederickson. Data structures for on-line updating of minimum spanning trees. In *Proc. of the 15 th ACM Symp. on Theory of Computing (STOC)*, pages 252–257, New York, NY, USA, 1983. ACM Press.

[GK98] P. Gupta and P. R. Kumar. Critical power for asymptotic connectivity. In *Proc. of the 37th IEEE Conference on Decision and Control*, volume 1, pages 1106–1110, 1998.

[GT01] M. Grossglauser and D. N. C. Tse. Mobility increases the capacity of ad-hoc wireless networks. In *Prof. of the 20th Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 1360–1369, 2001.

[HK97] M. Rauch Henzinger and V. King. Maintaining minimum spanning trees in dynamic graphs. In *Proc. of the 24 th Int. Colloq. on Automata, Languages and Programming (ICALP)*, pages 594–604, London, UK, 1997. Springer-Verlag.

[Hor07] C. Hornkamp. Mobile sensornetze in bürogebäuden. Master's thesis, Heinz Nixdorf Institute, University of Paderborn, 2007.

[KP95] E. Koutsoupias and C. H. Papadimitriou. On the *k*-server conjecture. *Journal of the ACM*, 42(5):971–983, 1995.

[Kru56] J.B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. In *Proceedings of the American Mathematics Society*, volume 7, pages 48–50, 1956.

[KTI+07] Y. Katayama, Y. Tomida, H. Imazu, N. Inuzuka, and K. Wada. Dynamic compass models and gathering algorithms for autonomous mobile robots. In *Proc. of the 14th Int. Colloq. on Structural Information and Communication Complexity (SIROCCO)*, volume 4474 of *Lecture Notes in Computer Science*, pages 274–288, 2007.

[Mat92] M. Mataric. Designing emergent behaviors: From local interactions to collective intelligence. In *Proc. of the International Conference on Simulation of Adaptive Behavior: From Animals to Animats*, volume 2, pages 432–441, 1992.

[Mat96] J. Matijevic. Mars pathfinder microrover - implementing a low cost planetary mission experiment. In *Proc. of the 2nd IAA International Conference on Low-Cost Planetary Missions*, pages IAA–L–0510, 1996.

[MMS90] M. S. Manasse, L. A. McGeoch, and D. D. Sleator. Competitive algorithms for server problems. *Journal of the ACM*, 11(2):208–230, 1990.

[MT01] B. Mohar and C. Thomassen. *Graphs on Surfaces*. The Johns Hopkins University Press, 2001.

[Mur04] R. R. Murphy. Rescue robotics for homeland security. *Communications of the ACM*, 47(3):66–68, 2004.

[NFP+03] H.G. Nguyen, N. Farrington, N. Pezeshkian, A. Gupta, and J. M. Spector. Autonomous communication relays for tactical robots. In *Proc. of the 11th International Conference on Advanced Robotics (ICAR)*, pages 35–40, 2003.

[NPGF04] H. G. Nguyen, N. Pezeshkian, A. Gupta, and N. Farrington. Maintaining communication link for a robot operating in a hazardous environment. In *Proc. of the 10th Int. Conf. on Robotics and Remote Systems for Hazardous Environments*. American Nuclear Society, 2004.

[NST00] P. Narvaez, K.-Y. Siu, and H.-Y. Tzeng. New dynamic algorithms for shortest path tree computation. In *IEEE/ACM Transactions on Networking*, volume 8, pages 734–746, 2000.

[NST01] P. Narvaez, K.-Y. Siu, and H.-Y. Tzeng. New dynamic spt algorithm based on a ball-and-string model. In *IEEE/ACM Transactions on Networking*, volume 9, pages 706–718, 2001.

[PR02] S. Pettie and V. Ramachandran. An optimal minimum spanning tree algorithm. *Journal of the ACM*, 49(1):16–34, 2002.

[Pri57] R.C. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36:1389–1401, 1957.

[RSS05] J. Remy, A. Souza, and A. Steger. On an online spanning tree problem in randomly weighted graphs. In *Combinatorics, Probability and Computing*, 2005.

[SB03] P. Santi and D. M. Blough. The critical transmitting range for connectivity in sparse wireless ad hoc networks. *IEEE Transactions on Mobile Computing*, 2(1):25–39, 2003.

[Sei01] S. S. Seiden. A general decomposition theorem for the k-server problem. In *Proc. of the 9th European Symp. on Algorithms (ESA)*, pages 86–97, 2001.

[ST85] D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.

[SY96] I. Suzuki and M. Yamashita. Distributed anonymous mobile robots — formation and agreement problems. In *Proc. of the 3rd Int. Colloq. on Structural Information and Communication Complexity (SIROCCO)*, volume 6 of *International Informatics Series*, pages 313–330. Carleton University Press, 1996.

[SY99] I. Suzuki and M. Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM Journal on Computing*, 28(4):1347–1363, 1999.

[WCLF02] P.-J. Wan, G. Calinescu, X.-Y. Li, and O. Frieder. Minimum-energy broadcasting in static ad hoc wireless networks. *Wireless Networks*, 8:607–617, 2002.

[XCZ+04] B. Xiao, J. Cao, Q. Zhuge, Z. Shao, and E. Hsing-Mean Sha. Dynamic update of shortest path tree in ospf. In *ISPAN*, pages 18–23, 2004.

[XK04] F. Xue and P. R. Kumar. The number of neighbors needed for connectivity of wireless networks. *Wireless Networks*, 10(2):169–181, 2004.

[Yao77] A. C.-C. Yao. Probabilistic computation: towards a uniform measure of complexity. In *Proc. of the 18th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 222–227, 1977.

[YCM+04] A. Young, J. Chen, Z. Ma, A. Krishnamurthy, L. Peterson, and R.Y. Wang. Overlay mesh construction using interleaved spanning trees. In *IEEE INFOCOM*, pages 396–407, 2004.

[You02] N. E. Young. On-line file caching. *Algorithmica*, 33(3):371–383, 2002.

Appendix

Go-To-The-Middle Correctness

Proof of Eq. (3.3). Let $x, y \in [-1, 1]$. We aim at showing that

$$\sqrt{(1-x^2)(1-y^2)} + x \cdot y \ \leq 1 \, .$$

It holds

$$\frac{\partial\left(\sqrt{(1-x^2)(1-y^2)} + x \cdot y\right)}{\partial x} = y - \frac{x}{\sqrt{1-x^2}} \cdot \sqrt{1-y^2} \, .$$

The derivative has only one zero for $x = y$. One can easily verify that this is a maximum as

$$y - \frac{x}{\sqrt{1-x^2}} \cdot \sqrt{1-y^2} > 0 \qquad \text{for } x < y$$
$$y - \frac{x}{\sqrt{1-x^2}} \cdot \sqrt{1-y^2} < 0 \qquad \text{for } x > y.$$

Therefore

$$\sqrt{(1-x^2)(1-y^2)} + x \cdot y \leq \left(1-x^2\right) + x^2 = 1 \, .$$

■

Upper Bounding the Binomial Distribution

In this section we will upper bound the probability distribution function of the binomial distribution. For ease of notation, let $B(k) = \binom{n}{k} \cdot p^k \cdot (1-p)^{n-k}$, i.e. $B(k)$ is the probability of obtaining k successes in n bernoulli trials with success probability p. We want to show the following lemma.

Lemma A.1. *Let $np \geq 2$ and $p \geq \frac{1}{2}$. Then it holds*

$$\max_{k=0,\ldots,n} B(k) \leq \frac{1}{\sqrt{2np^4}} .$$

We start with providing two simple lemmas about the binomial distribution.

Lemma A.2. *The function $B(k)$ assumes its maximum for $k = \lfloor (n+1) \cdot p \rfloor$.*

Proof. Simply plugging in the definition of $B(k)$ and reducing we obtain

$$\frac{B(k+1)}{B(k)} = \frac{(1-p) \cdot (k+1)}{p \cdot (n-k)} . \tag{A.1}$$

We have $B(k+1) > B(k)$ if and only if $\frac{(1-p)\cdot(k+1)}{p\cdot(n-k)} > 1$. Therefore, $B(k+1) > B(k)$ if and only if $k+1 < (n+1) \cdot p$. Similarly, $B(k+1) < B(k)$ if and only if $k+1 > (n+1) \cdot p$. Therefore, there exists a single maximum at $k = \lfloor (n+1) \cdot p \rfloor$. ■

Lemma A.3. *Let $np \geq 2$ and $p \leq \frac{1}{2}$. Then it holds*

$$\frac{\min\{B(\lfloor np \rfloor), B(\lceil np \rceil)\}}{p} \geq B(\lfloor (n+1) \cdot p \rfloor) .$$

Proof. Let us first show that

$$B(\lfloor np \rfloor) \cdot \frac{1}{p(1-p)} \geq B(\lfloor (n+1) \cdot p \rfloor) .$$

If $\lfloor np \rfloor = \lfloor (n+1) \cdot p \rfloor$ then the inequality follows trivially. Otherwise $\lfloor np \rfloor = \lfloor (n+1) \cdot p \rfloor - 1$. Then $\lfloor (n+1) \cdot p \rfloor \geq np$ and from Eq. (A.1) we have

$$\frac{B(\lfloor (n+1) \cdot p \rfloor)}{B(\lfloor (n+1) \cdot p \rfloor - 1)} = \frac{p \cdot (n - \lfloor (n+1) \cdot p \rfloor + 1)}{(1-p)(\lfloor (n+1) \cdot p \rfloor)} \leq \frac{p \cdot (n - np + 1)}{(1-p) \cdot np} .$$

By $p \leq \frac{1}{2}$ we have $1 - p \geq p$. As $np \geq 1$ we have

$$\frac{p \cdot (n - np + 1)}{(1-p) \cdot np} \leq \frac{1}{1-p} \leq \frac{1}{p} .$$

Now we turn to the second inequality, showing that

$$B(\lceil np \rceil) \cdot \frac{1}{p(1-p)} \geq B(\lfloor (n+1) \cdot p \rfloor) .$$

If $\lceil np \rceil = \lfloor (n+1) \cdot p \rfloor$ then the inequality follows trivially. Otherwise $\lceil np \rceil = \lfloor (n+1) \cdot p \rfloor + 1$. Once again using Eq. (A.1) we obtain

$$\frac{B(\lfloor (n+1) \cdot p \rfloor)}{B(\lfloor (n+1) \cdot p \rfloor + 1)} = \frac{(1-p) \cdot (\lfloor (n+1) \cdot p \rfloor + 1)}{p \cdot (n - \lfloor (n+1) \cdot p \rfloor)} \leq \frac{(1-p) \cdot (np + p + 2)}{p \cdot (n - np - p)} .$$

The latter term is smaller than $1/p$ if $n \geq 2/(1-p)$ which is true for $np \geq 2$ and $p \leq 1 - p$. ■

The function $B(k)$ is defined only for integer k and $k \geq 0$. Let us extend $B(k)$ to $\mathcal{B}(k)$ defined on positive real values by replacing all factorials in its definition with the Γ function. Define

$$b(x) = \frac{\Gamma(n+1)}{\Gamma(x+1) \cdot \Gamma(n-x+1)} .$$

Therefore we have

$$\mathcal{B}(x) = b(x) \cdot p^x \cdot (1-p)^{n-x} .$$

Obviously $B(k) = \mathcal{B}(k)$ for all integer k, as for all integer k it holds $b(k) = \binom{n}{k}$. We will now show that b(x) behaves well in a certain way.

Lemma A.4. *For any $x > 0$ we have*

$$b(x) \geq \min \{b(\lfloor x \rfloor), b(\lceil x \rceil)\} .$$

Proof. We will show that the function $b(x)$ has at most one extremum for $x > 0$ and if this extremum exists then it is a maximum. Recall that

$$b(x) = \frac{\Gamma(n+1)}{\Gamma(x+1) \cdot \Gamma(n-x+1)} .$$

The enumerator is obviously a constant, therefore we only have to take care that the term $\Gamma(x+1) \cdot \Gamma(n-x+1)$ has at most one extremum, and if one exists then it is a minimum. Differentiating we obtain

$$\frac{d\Gamma(x+1) \cdot \Gamma(n-x+1)}{dx} = \Gamma(x+1) \cdot \Gamma(n-x+1) \cdot (\psi(x+1) - \psi(n-x+1)) .$$

The latter term can become zero only due to $f(x) = \psi(x+1) - \psi(n-x+1) = 0$ as $\Gamma(x)$ is clearly positive for $x > 0$. We investigate the $\psi(y)$ function, which can be defined as

$$\psi(y) = \int_0^\infty \left(\frac{e^{-t}}{t} - \frac{e^{-yt}}{1-e^{-t}} \right) dt .$$

For any given $t \geq 0$ and any $\epsilon > 0$ we have

$$\left(\frac{e^{-t}}{t} - \frac{e^{-yt}}{1-e^{-t}}\right) \leq \left(\frac{e^{-t}}{t} - \frac{e^{-(y+\epsilon)t}}{1-e^{-t}}\right),$$

and thereby clearly $\psi(y)$ is a non-decreasing function. Therefore $f(x)$ is a non-decreasing function and can assume the zero value at most once. Assume that $f(y) = 0$. Then clearly $f(y-\epsilon) < 0$ and $f(y+\epsilon) > 0$ and therefore $\Gamma(x+1) \cdot \Gamma(n-x+1)$ has a minimum at point $x = y$.

For the sake of contradiction assume that there exists such $x > 0$, so that $b(x) < \min(b(\lfloor x \rfloor), b(\lceil x \rceil))$. Let the function $b(y)$ assume its minimum value in the interval $(\lfloor x \rfloor, \lceil x \rceil)$ at point y_m. Then obviously y_m is a local minimum. As by our earlier observations, $b(x)$ does not have any minima. This leads to a contradiction. ■

The following result is well-known as the Stirling's Approximation for the factorial function. We can approximate the Gamma function in a similar matter, as shown in [Art64].

Theorem A.5. *For $x \geq 0$ it holds $\Gamma(x) = \sqrt{2\pi \cdot x} \cdot \left(\frac{x}{e}\right)^x \cdot e^{\kappa_x}$, where $0 < \kappa_x < \frac{1}{12x}$.*

We are now ready to provide the proof of our main lemma.

Proof of Lemma A.1. By Lemma A.2 we know that $B(k)$ assumes its maximum at $k = \lfloor (n+1) \cdot p \rfloor$. By Lemma A.3, the definition of $\mathcal{B}(k)$ and Lemma A.4 we have

$$\begin{aligned}
\max_{k=0,\ldots,n} B(k) &\leq \frac{1}{p} \cdot \min\{B(\lfloor np \rfloor), B(\lceil np \rceil)\} \\
&= \frac{1}{p} \cdot \min\{\mathcal{B}(\lfloor np \rfloor), \mathcal{B}(\lceil np \rceil)\}. \\
&\leq \frac{1}{p} \cdot \mathcal{B}(np)
\end{aligned}$$

We still have to find an appropriate upper bound for $\mathcal{B}(np)$. Plugging in the approximation from Theorem A.5 and using $n \geq 1$ and $p \leq 1-p$ we obtain

$$\mathcal{B}(np) \leq \frac{\exp\left(\frac{1}{12}n\right)}{\sqrt{2\pi \cdot np \cdot (1-p)}} \leq \frac{1}{\sqrt{np \cdot (1-p)}} \leq \frac{1}{\sqrt{2np^2}}.$$

■