

Path Planning with spatial and temporal Constraints

Dissertation

Zur Erlangung des Doktorgrades (Dr. rer. nat.)
der Mathematisch-Naturwissenschaftlichen Fakultät
der Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

Florian Berger

aus Hamburg

Bonn, September 2010

Angefertigt mit der Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät
der Rheinischen Friedrich-Wilhelms-Universität Bonn

Gutachter: Prof. Dr. Rolf Klein, Universität Bonn
Prof. Dr. Norbert Blum, Universität Bonn

Tag der mündlichen Prüfung: 19. Januar 2011

Diese Dissertation ist auf dem Hochschulschriftenserver der ULB Bonn unter
http://hss.ulb.uni-bonn.de/diss_online elektronisch publiziert.

Erscheinungsjahr: 2011

Abstract

In this thesis we consider different problems arising in the context of planning the movement of objects. The common aspect of these problems is the interaction of spatial and temporal constraints.

The three main problems analyzed are:

- The first problem is a pursuit-evasion problem where some *lions* have the task to clear a grid graph whose nodes are initially contaminated. The contamination spreads one step per time unit in each direction not blocked by a lion. A vertex is cleared from its contamination whenever a lion moves to it. Brass *et al.* [21] showed that $\frac{n}{2}$ lions are not enough to clear the $n \times n$ -grid. We consider the same problem in dimension $d > 2$ and prove that $\Theta(n^{d-1}/\sqrt{d})$ lions are necessary and sufficient to clear the n^d -grid. Furthermore, we consider a 2-dimensional problem variant where the movement of the lions is not restricted to the edges of the grid. Lions are allowed to jump from grid vertices to non-adjacent grid vertices. For this problem variant we can show that the number of lions needed to clear the $n \times n$ -grid is at least $\frac{n}{2}$ and at most $\frac{n}{2} + O(\sqrt{n})$.
- The second main problem is to determine suitable meeting times and locations for a group of *participants* wishing to schedule a new meeting subject to already scheduled meetings possibly held at a number of different locations. Each participant must be able to reach the new meeting location, attend for the entire duration, and reach the next meeting location on time. We apply the concept of LP-type problems which leads to a randomized algorithm with expected linear running time. We also analyze several variants of the original problem and provide lower bounds for their solutions.
- The third main problem is the path planning for a *traveller* in an environment which contains moving carriers. While using carrier C , the traveller can walk on C at innate speed $v \geq 0$ in any direction, like a passenger on board a vessel. Whenever his current position on C is simultaneously contained in some other carrier C' , the traveller can change from C to C' , and continue his tour by C' . Given initial positions of the carriers and of a start position s and a goal position g , is the traveller able to reach g starting from s ? If so, what minimum travel time can be achieved?

In dimension 8 and higher, the problem is undecidable. An interesting case is in dimension 2. We prove that the problem is NP-hard, even if all carriers are vertical line segments. It turns out that an s -to- g path of finite duration may require an infinite number of carrier changes. Despite this difficulty, we can show that the two-dimensional problem is decidable if the carriers are lines or line segments. In addition, we provide a pseudo-polynomial approximation algorithm for the case that the carriers are bounded.

Acknowledgments

First of all, I would like to thank my advisor, Prof. Rolf Klein, for introducing me to the field of computational geometry, giving me the opportunity to write this thesis, and for many fruitful discussions. I am thankful to him and my other co-authors and colleagues.

I want to mention

Alexander Gilbers, Dr. Ansgar Grüne, Daniel Jung, Dr. Tom Kamphans, Martin Köhler, Dr. Elmar Langetepe, Prof. Doron Nussbaum, Rainer Penninger, Prof. Jörg-Rüdiger Sack and Dr. Jiehua Yi.

Also I would like to thank Antje Bertram and Mariele Knepper for assistance with administrative tasks.

Additionally, I would like to acknowledge with thanks Prof. Daniel Bienstock, Henry Bottomley, Dr. Thomas Böhme, Prof. Frank Dehne, Prof. Adrian Dumitrescu, Prof. David Eppstein, Prof. Fedor V. Fomin, Dr. Jérôme Galtier, Prof. David Kirkpatrick, Prof. Imre Leader, Friedrich Regen, Prof. Günter Rote, Jakob Söhl, Prof. Dimitrios M. Thilikos and Prof. Emo Welzl for valuable discussions or helpful advice on known results.

I am especially thankful to Prof. Jörg-Rüdiger Sack. He offered me the opportunity for a research stay abroad.

Above all, warmest thanks to my parents, Irene and Uwe Berger, for their constant encouragement and continuous support.

Thanks to all of you!

Contents

1	Introduction	1
1.1	Problems in path planning	1
1.2	Overview of this thesis	2
1.3	Related work	4
2	How many Lions are needed to clear a Grid?	7
2.1	Introduction	7
2.2	Definitions	9
2.3	Results in dimension 2	10
2.4	Results on d -dimensional grids	16
2.4.1	Upper bound	16
2.4.2	Lower bound	20
2.4.3	Asymptotic estimate	21
2.5	Flying lions in the 2-dimensional grid	22
2.6	Conclusion	26
3	Meeting Scheduling respecting Time and Space	27
3.1	Introduction	27
3.2	Problem definition	28
3.3	Geometric interpretation of the problem	29
3.4	LP-approach	30
3.4.1	General framework of LP-type problems	31
3.4.2	Preparations for applying the general framework	31
3.4.3	Application of the general framework	34
3.4.4	Allowing participants to travel at different speeds	36

3.5	Allowing k participants to be absent	38
3.6	Maximizing the number of participants	40
3.7	Multiple meeting locations	41
3.8	More than 2 previously scheduled meetings	41
3.9	Conclusion	44
4	Traveller's Problem	45
4.1	Introduction	45
4.1.1	Problem description	47
4.1.2	Surprising properties of traveller paths	49
4.1.3	Overview of the results of this chapter	53
4.2	Subintervals with finitely many carrier changes	53
4.3	Higher dimensions	55
4.4	Dimension two	65
4.4.1	NP-hardness	65
4.4.2	Decidability	67
4.4.2.1	Analyzing paths by means of the graph G	67
4.4.2.2	Propagation of reachable points	71
4.4.2.3	Treatment of Zeno cycles	75
4.4.3	Pseudo-polynomial approximation	81
4.5	Uncountable many paths and the Cantor set	84
4.6	Conclusion	86
	List of Figures	87
	Bibliography	89

Chapter 1

Introduction

1.1 Problems in path planning

Designing suitable paths for movable objects is a crucial task in many scenarios. Motion planning involves such diverse aspects as computing collision-free paths among possibly moving obstacles, coordinating the motions of several robots, and planning sliding and pushing motions to achieve precise relations among objects. An overview of motion planning is given in the books by Latombe [46] and LaValle [47].

A simple example for one basic problem in motion planning is to find the shortest path from a source to a destination in a scene given by a simple polygon [49].

Many different variations of such problems are possible which leads to a rich number of interesting problems. Let us mention only a few possible variants:

- Are all information given in advance or does the robot achieve further information while moving? Corresponding problems are called *offline* in the first case and *online* in the second case. In this thesis our focus is on the offline problems.
- How is the movement of the robot modeled?
- Is the shape of the robot given by a point or by a more complex object?
- Are obstacles changing during time?
- Are there any devices (such as escalators or ferries) supporting the movement of the robot?
- Does the movement of the robot change the surrounding scene as it is the case in the well known game “Sokoban” [26]?
- Is the number of robots whose paths have to be planned greater than one?

In case of multiple robots additional arising questions might be:

- Is there any influence between the movement of the different robots?
- Is it required that the robots keep a certain distance from each other?
- Or, in the contrary, do they have to meet at a common point?

Possible objective functions for the motion planning could be:

- Minimize the travel time until the (last) robot reaches its goal.
- If the robots change the surrounding scene, can they achieve a certain goal configuration? If yes, what is the minimum time needed to do so?
- Maximize the safety distance between the robots and/or the obstacles.
- Maximize the number of robots which can be moved simultaneously.
- If the task is that all robots meet at a common point, how to choose this point?

The above listing does not at all claim to be complete. However, it allows combinations leading to the three main problems analyzed in this thesis.

1.2 Overview of this thesis

This work considers three main problems. These main problems fit into the context “Path Planning with spatial and temporal Constraints”. Apart from this, they are only loosely connected. In particular, it is possible to read the corresponding chapters in arbitrary order.

In Chapter 2 we consider a pursuit-evasion problem where some *lions* have the task to clear a grid graph whose nodes are initially contaminated. The contamination spreads one step per time unit in each direction not blocked by a lion. A vertex is cleared from its contamination whenever a lion moves to it. Brass *et al.* [21] showed that $\frac{n}{2}$ lions are not enough to clear the $n \times n$ -grid. We consider the same problem in dimension $d > 2$ and a modified version in dimension 2.

For dimension $d > 2$, we present a solution which lets the lions sweep the grid vertices layer by layer, where the r -th layer is defined as the set of vertices having L_1 -distance r from the origin. The number of lions needed to clear the grid this way can be upper bounded by two times the size of the middle layer of the grid. On the other hand, we can also show that any successful clearing strategy requires a number of lions which is at least $\frac{1}{6}$ times the size of the middle layer.

Furthermore, we consider a 2-dimensional problem variant where the movement of the lions is not restricted to the edges of the grid. Lions are allowed to fly from grid vertices to non-adjacent grid vertices. For this problem variant we can show that the number of lions needed to clear the $n \times n$ -grid is at least $\frac{n}{2}$ and at most $\frac{n}{2} + O(\sqrt{n})$. We published these results first in [13, 12].

Chapter 3 deals with a meeting scheduling problem. We consider the problem of determining suitable meeting times and locations for a group of *participants* wishing to schedule a new meeting subject to already scheduled meetings possibly held at a number of different locations. Each participant must be able to reach the new meeting location, attend for the entire duration, and reach the next meeting location on time. In contrast to both other main problems of this thesis, it turns out very helpful for this problem to think of the time as an additional geometric dimension. Then the problem can be stated as finding a vertical segment of maximum length in the intersection of certain cones. Our approach to solve it uses the concept of LP-type problems [73]. We establish that our meeting scheduling problem can be regarded as an LP-type problem whose combinatorial dimension equals 4. This observation leads to a randomized algorithm with expected running time $O(n)$ improving the previously known $O(n \log n)$ result by [43]. Our $O(n)$ solution also applies for the case that the participants are allowed to travel at different speeds. We published these results first in [16, 17].

Additionally, we investigate some variants of this problem. We show that the problem to determine a meeting of maximum duration for all except k participants is *3SUM-hard*, see [32] for the definition of the class of 3SUM-hard problems, and give an $O(n^3 \log n)$ algorithm for this problem variant. Furthermore, we prove the problem to find the optimum meeting for n participants at k different meeting locations to be NP-hard. On the other hand, if only one meeting location is required, but every participant has up to at most k meetings already scheduled, then a solution with running time $O(kn^2)$ is possible.

In Chapter 4 we introduce a *Traveller's Problem*. A traveller is planning a tour from some start position s to a goal position g in d -dimensional space. Transportation is provided by n carriers. Each carrier is a convex object that results from intersecting finitely many closed half-spaces; it moves at constant speed along a line. Different carriers may be assigned different velocity vectors. While using carrier C , the traveller can walk on C at innate speed $v \geq 0$ in any direction, like a passenger on board a vessel. Whenever his current position on C is simultaneously contained in some other carrier C' , the traveller can change from C to C' , and continue his tour by C' .

Given initial positions of the carriers and of s and g , is the traveller able to reach g starting from s ? If so, what minimum travel time can be achieved?

We will establish that the complexity of Traveller's Problem arises from two different points of view. The first one is an *analytical complexity*, caused by the following surprising fact. There are very simple scenes in two-dimensional space where the goal

can be reached in finite time, but only by an infinite number of carrier changes. The situation is even worse in three-dimensional space where the set of carrier change time points may require infinitely many accumulation points for the traveller to reach his goal. The other big challenge besides the analytical complexity is the *combinatorial complexity* of the problem. The hardness results mentioned in the following still hold, even if the number of carrier changes is restricted to be finite.

In dimension 8 and higher, Traveller's Problem is undecidable. Our undecidability proof uses a recent result by Bell and Potapov [10] who showed that the problem whether a given rational point in the plane can be mapped to another one by a finite product of affine mappings from a given set is undecidable.

An interesting case is in dimension 2. Using a reduction from the PARTITION problem, we prove that the two-dimensional Traveller's Problem is NP-hard, even if all carriers are vertical line segments. Despite the difficulty of infinitely many carrier changes, we can show that the two-dimensional problem is decidable if all carriers are lines or line segments. In addition, we provide a pseudo-polynomial approximation algorithm. This algorithm works by discretising time and space. It outputs a path in a relaxed model where the traveller's innate speed and the extensions of the carriers are slightly increased.

We presented our discoveries about Traveller's Problem first in [14, 15].

Each Chapter 2, 3 and 4 starts with an introduction which summarizes its content.

1.3 Related work

Most related work is mentioned in the introductions of the following chapters. The purpose of this section is to mention some other known results on motion planning questions resulting from the listing of variants on pages 1 and 2.

Lee and Preparata [49] showed that the shortest path between two points in a simple polygon with n edges can be computed in time $O(n)$. Guibas and Hershberger [37] provided an algorithm to preprocess a simple polygon such that afterwards the length of the shortest path between two arbitrary points of the polygon can be computed in time $O(\log n)$. Even for polygons with holes, a logarithmic query time can be achieved if one accepts a high preprocessing time and high space usage, see [24].

Schwartz and Sharir [70] established that general motion planning problems are decidable, if they can be answered by analyzing the connectedness of a semialgebraic configuration space.

Hopcroft *et al.* [38] proved that the motion planning problem for a collection of disjoint two-dimensional rectangular objects constrained to move within a two-dimensional rectangular box is PSPACE-hard.

Canny and Reif [22] showed that the problem to find a shortest path between two

points amid polyhedral obstacles in 3-dimensional space is NP-hard. The situation is different if we are given a polyhedral surface and seek a path which is constrained to lie on this surface. For this case Mitchell *et al.* [57] presented an algorithm to determine the length of a shortest path in logarithmic query time after $O(n^2 \log n)$ preprocessing. For their algorithm, they developed the *continuous Dijkstra paradigm*, which we will also apply to solve a special case of Traveller's Problem.

Our next example is the well known computer game "Sokoban". This game is a transport puzzle, in which the player pushes boxes around a maze, viewed from above, and tries to put them in designated locations. Despite its simple description, Culberson proved Sokoban to be PSPACE-complete [26]. It is a common aspect of many motion planning problems that they have high complexity though they seem to be very simple and innocent at first glance.

The user playing Sokoban is doing the motion planning, while his computer only checks whether his moves are allowed by the rules of the game and displays the result. Nowadays, many computer games themselves perform motion planning tasks. This brings us to one motivation to consider motion planning problems. In many computer games, entities need to move around in natural ways and plan their routes amidst obstacles and other moving entities. Motion-planning techniques that originate from robotics have been adapted and effectively applied in the development of computer games, see *e.g.* [59].

Finally, let us mention an example of an undecidable motion planning problem. Undecidability was shown for *frictional mechanical systems* by Reif and Sun [67]. For frictional mechanical systems, a set of objects in 3-dimensional space is given. Each surface patch is either frictional or sliding. The robot can move an object not only by grasping or pushing it, but also by using the friction between the object and the surrounding objects. The question is whether an initial configuration can be changed to a goal configuration.

Chapter 2

How many Lions are needed to clear a Grid?

2.1 Introduction

Pursuit-evasion problems have a long history in mathematics and computer science, and many different models have been studied. At SoCG'07, Dumitrescu *et al.* [27] introduced a variant that has, apparently, not received much attention before.

There are two different ways to consider this problem. The first approach, used by Dumitrescu, is to regard it as a pursuit-evasion problem. Then, the situation is that we have a pride of lions prowling among the vertices and edges of a d -dimensional $n \times \dots \times n$ grid. If their paths are known in advance, is it possible to design a safe path for a man that avoids all lions, assuming that man and lion move simultaneously and at the same speed along the graph edges? To be more precise, let G denote the d -dimensional $n \times \dots \times n$ grid with vertex set V . A path π visiting $p \in V$ at time $t \in \{0, \dots, T-1\}$ may visit a direct neighbor q of p at time $t+1$, or remain at p . Two paths π_1, π_2 are said to avoid each other if they never occupy the same vertex at the same time t and, in their transition from t to $t+1$, never traverse the same grid edge from opposite sides. Now the problem is the following. What is the maximum number $k = k_d(n)$ such that for all possible sets of k “lion” paths with arbitrary length T in G one can construct a “man” path that avoids them all?

For such games there are many different denotations for the pursuer and the haunted in the literature. Besides *lion and man*, also *cop and robber* or *offline searcher and target* are common terms. However, we prefer to take the *lion and man* notation used by Dumitrescu.

The second approach to the problem above even makes the *man* unnecessary. Instead of the path of one man, let us consider the set $W(t)$ of all vertices where this man could be at time t . That is, $W(0)$ equals V minus the lions' start positions, and $W(t+1)$

consists of all vertices p of V that

- belong to $W(t)$ and are not visited by a lion at time $t + 1$, or
- are not occupied by a lion at time $t + 1$, and have a direct neighbor q in $W(t)$ that is not visited at time $t + 1$ by a lion coming directly from p .

Now, let us side with the lions! We may consider $W(t)$ as the set of locations that are, at time t , contaminated by some evil force that spreads one step per time unit in each direction not blocked by a lion. In this model the lions' task is to fight contamination. A lion clears a contaminated vertex by visiting it. Once the lion is gone, the vertex may become recontaminated, according to the rules stated above.

These definitions give rise to the following observation. A group of k lions is able to catch a man independently of how he moves if and only if they can shrink the set of contaminated vertices $W(t)$ until it becomes the empty set. With this interpretation, our problem can be stated as follows:

How many lions are needed to clear an initially contaminated $n \times \dots \times n$ grid?

This problem differs from the classical man-and-lion problem introduced by Rado, see Littlewood [51, pp. 114–117] or Alonso *et al.* [2], where one lion knows the position of the man and moves in continuous time and space to catch him. A recently analyzed problem variant where unmanned air vehicles are used for searching one or more evading targets moving in a predefined area is described in [3]. Our problem also differs from the classical graph search problem introduced by Parson [61], where the contamination is allowed infinite speed. The literature of graph searching is much too broad to be summarized here, see Bienstock [18] for a survey.

It is not clear how to adapt to our problem the proof of LaPaugh's [45] result that a searchable graph can be searched optimally without recontamination. Considering our problem on arbitrary graphs instead of grid graphs, Penninger [62] found a planar graph where indeed allowing recontamination reduces the number of necessary lions in our setting.

Our problem at hand also differs from the cop-and-robber games investigated by Nowakowski and Winkler [60], in that they consider the online situation where the lions can adapt their paths to the escape maneuvers of their prey.

Clearly, n lions are able to clear the 2-dimensional grid, by performing a left-to-right sweep in column formation. More generally, $O(n)$ lions are sufficient to clear any planar graph over n^2 vertices. Namely, due to Lipton and Tarjan [50] there exists a $c \cdot n$ vertex separator where one group of lions can be positioned, while the remaining subgraphs are recursively cleared one by one.

One could think it obvious that n lions are necessary to clear a two-dimensional $n \times n$ grid, on the belief that a line sweep is the best way to do so. By the same reasoning, one could conjecture that it takes n^{d-1} lions to decontaminate a d -dimensional grid of size n , because a hyperplane sweep seems the best possible way to do so. While the

two-dimensional situation is still open, the above conjecture for d -dimensional grids is wrong, as we shall show.

We introduce necessary formal definitions in Section 2.2, and prove in Section 2.3 that $\frac{n}{2}$ lions are unable to clear the two-dimensional $n \times n$ grid. This result [13] was obtained independently and simultaneously with the Brass *et al.* [21] group of researchers. It improves on a previous $O(\sqrt{n})$ lower bound by Dumitrescu *et al.* [27].

In Section 2.4, we first demonstrate how 8 lions, rather than $n^{d-1} = 3^2 = 9$, can clear the $3 \times 3 \times 3$ grid. This is a simple counterexample to the above conjecture that it takes n^{d-1} lions to decontaminate a d -dimensional grid of size n . More generally, we prove that the minimum number of lions needed to clear a d -dimensional grid of size n equals the number of grid vertices of L_1 -distance $\lfloor \frac{d(n-1)}{2} \rfloor$ to the origin, up to a constant factor. Since no closed formula for this number seems to be known, we employ a folk-theorem that establishes an asymptotic estimate by means of the central limit theorem. As a consequence, we obtain that the d -dimensional grid of size n can be cleared by $\Theta(\frac{n^{d-1}}{\sqrt{d}})$ many lions.

In Section 2.5, we consider a two-dimensional problem variant where the movement of the lions is not restricted to the grid. Lions are allowed to fly from grid vertices to non-adjacent grid vertices. For this problem variant, we can show that the number of lions needed to clear the $n \times n$ -grid is at least $\frac{n}{2}$ and at most $\frac{n}{2} + O(\sqrt{n})$.

We published our results first in [13, 12].

2.2 Definitions

We define $[n] := \{0, \dots, n-1\}$. Let $G_n^d = (V_n^d, E_n^d)$, or $G = (V, E)$, denote the integer grid of size n in dimension d . The vertex set V_n^d equals $[n]^d$ and the edge set E_n^d is given by the pairs of vertices having L_1 -distance 1 from each other. For any vertex $v \in V$ we denote the L_1 -distance from v to the origin 0 as $|v|$. The *closed neighborhood* of v is defined as $\mathcal{N}(v) := \{w \in V \mid |v - w| \leq 1\}$. Accordingly, for every $A \subseteq V$ we define the closed neighborhood of A as

$$\mathcal{N}(A) := \{v \in V \mid \exists w \in A : |v - w| \leq 1\}.$$

Paths π in G are parameterized by time. Vertex $\pi(t+1)$ may be equal to $\pi(t)$, or to one of its direct neighbors in G . For a given set of k lion paths π_j , let $W(t)$ denote the set of vertices contaminated at time t .

That is, $W(0)$ contains all vertices of G except the lions' start positions, and $W(t+1)$ consists of all vertices $p \notin \{\pi_1(t+1), \dots, \pi_k(t+1)\}$ that

- belong to $W(t)$, or
- have a direct neighbor q in $W(t)$ that is not visited at time $t+1$ by a lion coming directly from p .

Clearing question in G_n^d . Given k lion paths over $\{0, \dots, T\}$ in G_n^d , does there exist a time $t \in \{0, \dots, T\}$ such that $W(t) = \emptyset$? We say that G_n^d has *clearing number* $k_d(n)$, if

- for arbitrary large T and arbitrary $k_d(n) - 1$ lion paths, we have $W(t) \neq \emptyset$ for all $t \in \{0, \dots, T\}$, **and**
- there exist $k_d(n)$ lion paths and a time t such that $W(t) = \emptyset$.

Let us now consider k arbitrary paths $\pi_1, \dots, \pi_k : \{0, \dots, T\} \rightarrow V$. If, for some time $t \in \{0, \dots, T\}$, we have $v \notin W(t)$, then we call the vertex v *cleared* at time t . Figure 2.1 shows an example. Let $\mathcal{C}(t)$ denote the set of cleared vertices at time t .

As Dumitrescu *et al.* [27] observed for $d = 2$, it is easy to verify that

$$k_d(n) \leq n^{d-1}$$

holds, by sweeping the grid with a hyperplane manned with n^{d-1} lions.

There are different types of cleared vertices depending on their neighborhood. A cleared vertex $v \in \mathcal{C}(t)$ is a cleared *interior* vertex if all of its neighbors are also cleared, i.e., $\mathcal{N}(v) \subseteq \mathcal{C}(t)$. Otherwise it is a cleared *boundary* vertex. More generally, for any vertex set $C \subset [n]^d$ we define the set of *boundary vertices* as

$$\partial C := \{v \in C \mid \mathcal{N}(v) \cap \overline{C} \neq \emptyset\}$$

where $\overline{C} := [n]^d \setminus C$ denotes the complement of C .

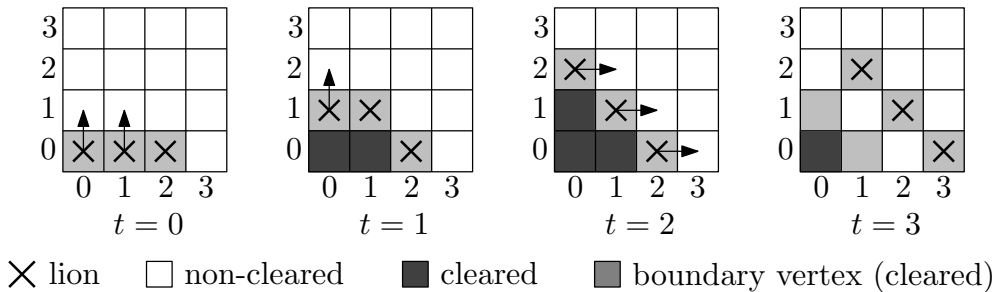


Figure 2.1: $k = 3$ lions try to decontaminate a 2-dimensional $n \times n$ -grid, where $n = 4$. (In this illustration the vertices are cells, and edges exist between neighbor cells.)

2.3 Results in dimension 2

In this section, we consider the two-dimensional case and prove the following result which was first shown in [21, 13].

Theorem 2.1. $\frac{n}{2}$ lions are not capable of clearing a 2-dimensional grid of size n . In other words, $k_2(n) > \frac{n}{2}$.

In the following the variable k always denotes the number of lions. As a first step we mention the following simple Lemma which holds because each lion can only clear one vertex within each step. The proof of Lemma 2.3 is slightly more involved. Observe that both, Lemma 2.2 and Lemma 2.3, hold for arbitrary graphs.

Lemma 2.2. *The number of cleared vertices cannot increase by more than k within one time step.*

Lemma 2.3. *If there are at least $2k$ boundary vertices in $\mathcal{C}(t)$, then the amount of cleared vertices cannot increase in the following step.*

$$|\partial\mathcal{C}(t)| \geq 2k \Rightarrow |\mathcal{C}(t+1)| \leq |\mathcal{C}(t)|.$$

Proof. We assume $|\partial\mathcal{C}(t)| \geq 2k$ for arbitrary, but fixed t . We define the set

$$C_1 := \mathcal{C}(t) \setminus \partial\mathcal{C}(t)$$

as the set of cleared interior vertices at time t . Notice that $|C_1| = |\mathcal{C}(t)| - |\partial\mathcal{C}(t)|$.

Let $v \in \partial\mathcal{C}(t)$ be a boundary vertex with i non-cleared neighbors. If there are at least i lions located on v at time t and these lions move in such a way that they use every edge leading to a non-cleared neighbor, then v remains cleared. In this case, we say that v is *protected from recontamination by leaving lions*. For example the vertex $(0,0)$ is protected by a leaving lion in the first step ($t=0$) of Figure 2.1.

We define the set C_2 as the set of grid vertices which:

- are occupied by a lion at time $t+1$, or
- are belonging to $\partial\mathcal{C}(t)$ and are protected from recontamination in the following step by leaving lions.

Both numbers, the number of vertices occupied by a lion at time $t+1$ as well as the number of vertices protected from recontamination by leaving lions, are at most k . We conclude that $|C_2| \leq 2k$. Using $\mathcal{C}(t+1) = C_1 \cup C_2$, we obtain $|\mathcal{C}(t+1)| = |C_1 \cup C_2| \leq |C_1| + |C_2| = |\mathcal{C}(t)| - |\partial\mathcal{C}(t)| + |C_2| \leq |\mathcal{C}(t)|$. \square

Next, we will establish that every set $C \subset V_n^2$ of approximately $\frac{n^2}{2}$ cleared vertices has at least n boundary vertices. This isoperimetric inequality is by Bollobás and Leader [20]. For convenience, we include a direct proof in dimension 2. Its Lemma 2.4 and an analogue to Lemma 2.5 were also used by Galtier [33].

We introduce the *fall-down transformation* (Bollobás and Leader [20] call those transformations *compressions* and Galtier [33] calls it *pushing process*.) which enables us

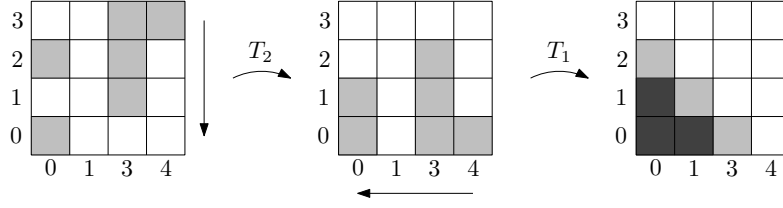


Figure 2.2: The fall-down transformation does not increase the number of boundary cells (=vertices).

to concentrate on situations where the cleared vertices are spread in a monotone way, cf. Figure 2.2. The idea is to turn on a kind of gravity which tows all cleared vertices downward towards the 0-level of a particular coordinate i , and to execute this transformation for every coordinate $i \in \{1, \dots, d\}$.

More formally, let $C \subseteq [n]^d$ be an arbitrary subset of the grid-vertices. Let $i \in \{1, \dots, d\}$ be arbitrary, and let

$$n_{i,C}(v_1, \dots, v_{i-1}, v_{i+1}, \dots, v_d) := |\{(v_1, \dots, v_{i-1}, a, v_{i+1}, \dots, v_d) \in C \mid a \in [n]\}|$$

denote the number of vertices of C in the column over the vertex $(v_1, \dots, v_{i-1}, 0, v_{i+1}, \dots, v_d)$. For simplicity, we will often omit the C in $n_{i,C}(V)$. The *fall-down transformation with respect to coordinate i* is defined by

$$T_i(C) := \{(v_1, \dots, v_d) \mid v_i < n_i(v_1, \dots, v_{i-1}, v_{i+1}, \dots, v_d)\}.$$

The general *fall-down transformation* is a concatenation of all such transformations,

$$T : \mathcal{P}([n]^d) \rightarrow \mathcal{P}([n]^d), \quad T(C) := T_1 \circ T_2 \circ \dots \circ T_d(C).$$

Note that changing the order in this concatenation can alter the result but not the monotonicity of the result. A set $C \subseteq [n]^d$ is said to be *i -monotone*, $i \in \{1, \dots, d\}$, if

$$\begin{aligned} \forall (v_1, \dots, v_{i-1}, v_i, v_{i+1}, \dots, v_d) \in C : v_i > 1 \\ \Rightarrow (v_1, \dots, v_{i-1}, v_i - 1, v_{i+1}, \dots, v_d) \in C. \end{aligned}$$

The set C is *monotone* if it is i -monotone for every $i \in \{1, \dots, d\}$.

Lemma 2.4. *The result of the fall-down transformation is monotone.*

Proof. By definition it is clear that the result of each T_i is i -monotone.

It remains to show that for every $i, j \in \{1, \dots, d\}$, $i \neq j$, and for every $C \subseteq [n]^d$ which is j -monotone the transformed set $T_i(C)$ is still j -monotone. To this end, let us assume that $C \subseteq [n]^d$ is j -monotone, $i \neq j$.

Let $v = (v_1, \dots, v_d) \in [n]^d$ be an arbitrary vertex satisfying $v_j > 0$. Then, we have

$$\begin{aligned}
& n_i(v_1, \dots, v_{i-1}, v_{i+1}, \dots, v_d) & (2.1) \\
& = |\{(v_1, \dots, v_{i-1}, a, v_{i+1}, \dots, v_{j-1}, v_j, v_{j+1}, \dots, v_d) \in C \mid a \in \{1, \dots, d\}\}| \\
& \leq |\{(v_1, \dots, v_{i-1}, a, v_{i+1}, \dots, v_{j-1}, v_j - 1, v_{j+1}, \dots, v_d) \in C \mid a \in \{1, \dots, d\}\}| \\
& = n_i(v_1, \dots, v_{i-1}, v_{i+1}, \dots, v_{j-1}, v_j - 1, v_{j+1}, \dots, v_d)
\end{aligned}$$

because the j -monotonicity of C implies that for every $(v_1, \dots, v_{i-1}, a, v_{i+1}, \dots, v_d) \in C$ also the vertex $(v_1, \dots, v_{i-1}, a, v_{i+1}, \dots, v_{j-1}, v_j - 1, v_{j+1}, \dots, v_d)$ belongs to C .

We want to prove

$$\forall (v_1, \dots, v_d) \in T_i(C) : v_j > 0 \Rightarrow (v_1, \dots, v_{j-1}, v_j - 1, v_{j+1}, \dots, v_d) \in T_i(C).$$

We assume that the preconditions are fulfilled. By (2.1) we can conclude

$$\begin{aligned}
v_i & \stackrel{v \in T_i(C)}{\leq} n_i(v_1, \dots, v_{i-1}, v_{i+1}, \dots, v_{j-1}, v_j, v_{j+1}, \dots, v_d) \\
& \stackrel{(2.1)}{\leq} n_i(v_1, \dots, v_{i-1}, v_{i+1}, \dots, v_{j-1}, v_j - 1, v_{j+1}, \dots, v_d).
\end{aligned}$$

By definition of $T_i(C)$ this proves $(v_1, \dots, v_{j-1}, v_j - 1, v_{j+1}, \dots, v_d) \in T_i(C)$. \square

Lemma 2.5. *The number of boundary vertices does not increase by the fall-down transformation.*

Proof. Obviously it suffices to prove the statement for each T_i . First, we figure out how many boundary vertices exist in each column of $T_i(C)$ where $i \in \{1, \dots, d\}$ and $C \subseteq [n]^d$ are arbitrary. Let $v \in [n-1]^d$ be an arbitrary vertex. We consider the column on top of $(v_1, \dots, v_{i-1}, 0, v_i, \dots, v_{d-1})$, i.e. the set

$$\{(v_1, \dots, v_{i-1}, a, v_i, \dots, v_{d-1}) \mid a \in [n]\}.$$

It contains $n_i(v) = n_{i, T_i(C)}(v) = n_{i, C}(v)$ many vertices of $T_i(C)$ and the same number of vertices of C .

It is not difficult to see that the number of boundary-vertices of $T_i(C)$ in this column equals

$$\max \left(\max_{w \in \mathcal{N}(v)} n_i(v) - n_i(w), \chi_{1 \leq n_i(v) \leq n-1} \right) \quad (2.2)$$

where

$$\chi_{1 \leq n_i(v) \leq n-1} := \begin{cases} 1 & \text{if } 1 \leq n_i(v) \leq n-1 \\ 0 & \text{otherwise} \end{cases}$$

First, let us consider the case where Expression (2.2) equals zero. Then, either we have $n_i(v) = 0$ or we have $n_i(v) = n$ and also $n_i(w) = n$ for every $w \in \mathcal{N}(v)$. In both cases the column of v does not contain any boundary vertices of C either.

Now suppose that Expression (2.2) does not equal zero, but still the maximum is attained by $\chi_{1 \leq n_i(v) \leq n-1}$. In this case the column of v is neither full nor empty, neither with respect to $T_i(C)$ nor with respect to C . Hence, there must also exist at least one boundary vertex of C in this column.

Finally, suppose the maximum is attained by $\max_{w \in \mathcal{N}(v)} n_i(v) - n_i(w)$. And let w be a neighbor vertex which maximizes $n_i(v) - n_i(w)$. In this case we must have $n_i(v) > n_i(w)$. The column of w contains exactly $n_i(w)$ vertices of C and the column of v contains exactly $n_i(v)$ vertices of C . Thus, at least for $n_i(v) - n_i(w)$ vertices of C in the v -column the corresponding neighbor in the w -column does not belong to C . They are boundary vertices of C .

We have shown that for each column the number of boundary vertices of C cannot be less than the number of boundary vertices of $T_i(C)$. This proves the claim. \square

Now, we are ready to prove the following Lemma.

Lemma 2.6. Any vertex set $C \subset V_n^2$ satisfying $\frac{n^2}{2} - \frac{n}{2} < |C| < \frac{n^2}{2} + \frac{n}{2}$ has at least n boundary vertices.

Proof. Due to Lemma 2.5 it suffices to prove the claim for monotone sets $C \subset V_n^2$. Note that in the two-dimensional setting, $n_1(j)$ denotes the number of vertices of C in the j -th row and $n_2(i)$ denotes the number of vertices of C in the i -th column. Because of the monotonicity we have

$$n_1(0) \geq n_1(1) \geq \dots \geq n_1(n-1) \quad \text{and} \quad n_2(0) \geq n_2(1) \geq \dots \geq n_2(n-1).$$

And clearly $\sum_{j=0}^{n-1} n_1(j) = \sum_{i=0}^{n-1} n_2(i) = |C|$ holds.

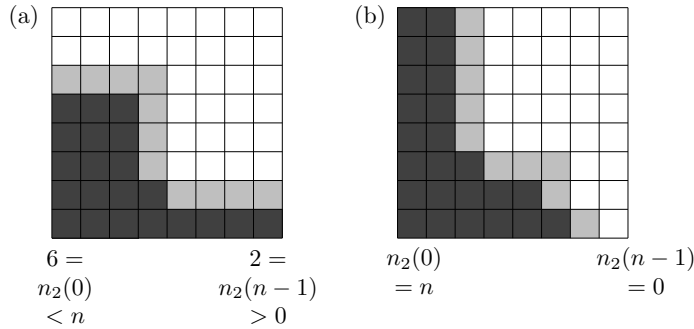


Figure 2.3: Two simple cases with at least n boundary vertices.

Consider Figure 2.3(a). If all the columns are neither completely full nor totally empty, i.e., $n_2(0) < n$ and $n_2(n-1) > 0$, then every column contains at least one boundary vertex, and the proof is complete. Otherwise, we have $n_2(0) = n$ or $n_2(n-1) = 0$.

We consider the first case, $n_2(0) = n$. Note that in this case $n_2(n-1) = 0$ would imply that all the rows are neither completely empty nor completely full and the proof would be complete, cf. Figure 2.3(b). Hence, it suffices to consider $n_2(0) = n$ and $n_2(n-1) > 0$.

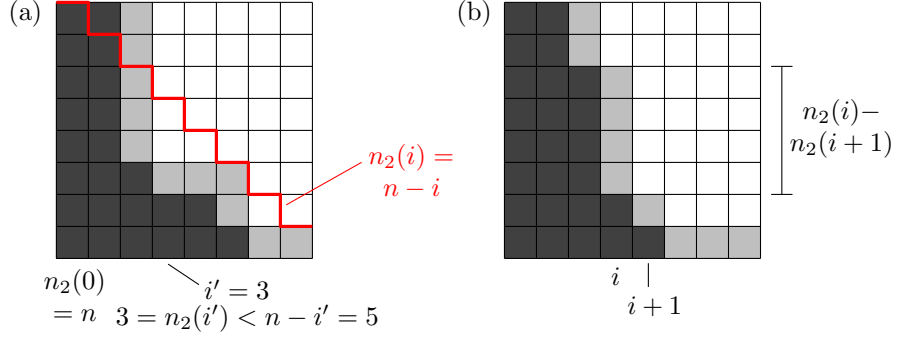


Figure 2.4: (a) There exists a column satisfying $n_2(i') < n - i'$. (b) Column i contains at least $n_2(i) - n_2(i+1)$ boundary vertices.

Consider Figure 2.4(a). There must be a column i' satisfying $n_2(i') < n - i'$. Otherwise we had

$$\begin{aligned} |C| &\geq \sum_{i=0}^{n-1} (n-i) = n^2 - \sum_{i=0}^{n-1} i = n^2 - \frac{n(n-1)}{2} \\ &= n^2 - \frac{n^2}{2} + \frac{n}{2} = \frac{n^2}{2} + \frac{n}{2}. \end{aligned}$$

For such a column i' we have

$$\sum_{i=0}^{i'-1} (n_2(i) - n_2(i+1)) = n_2(0) - n_2(i') \geq n - (n - i') + 1 = i' + 1.$$

However, each column $i \in \{0, \dots, i'-1\}$ contains at least $n_2(i) - n_2(i+1)$ boundary vertices, see Figure 2.4(b). Hence all these columns contain at least $i' + 1$ boundary vertices. And the remaining columns $i'+1, \dots, n-1$ contain at least $n - i' - 1$ boundary vertices. This completes the proof for the case $n_2(n) > 0$. The case $n_2(n-1) = 0$ can be treated analogously. \square

Now we are able to prove Theorem 2.1.

Proof. If $k = \lfloor \frac{n}{2} \rfloor$ lions were able to clear G_n^2 , they would have to extend the set of cleared vertices until $|\mathcal{C}(T)| = n^2$. By Lemma 2.2 we know that $|\mathcal{C}(t+1)| - |\mathcal{C}(t)| \leq k \leq \frac{n}{2}$ for every t . Hence, there had to be a time t such that $\frac{n^2}{2} - \frac{n}{4} \leq |\mathcal{C}(t)| \leq \frac{n^2}{2} + \frac{n}{4}$ and $|\mathcal{C}(t+1)| > |\mathcal{C}(t)|$. But, by Lemma 2.6, there would be at least n boundary vertices of $\mathcal{C}(t)$ at time t , and Lemma 2.3 shows that $|\mathcal{C}(t+1)| \leq |\mathcal{C}(t)|$, a contradiction. \square

2.4 Results on d -dimensional grids

2.4.1 Upper bound

As mentioned in the introduction, one could conjecture $k_d(n) = n^{d-1}$. However, this does not even hold for $d = 3$, see Figure 2.5. The movement can be done such that only one lion changes its location within each time step. All remaining lions can stay at their current positions and protect the cleared vertices. Even if the moving lion needs several steps to reach its destination, no problem arises.

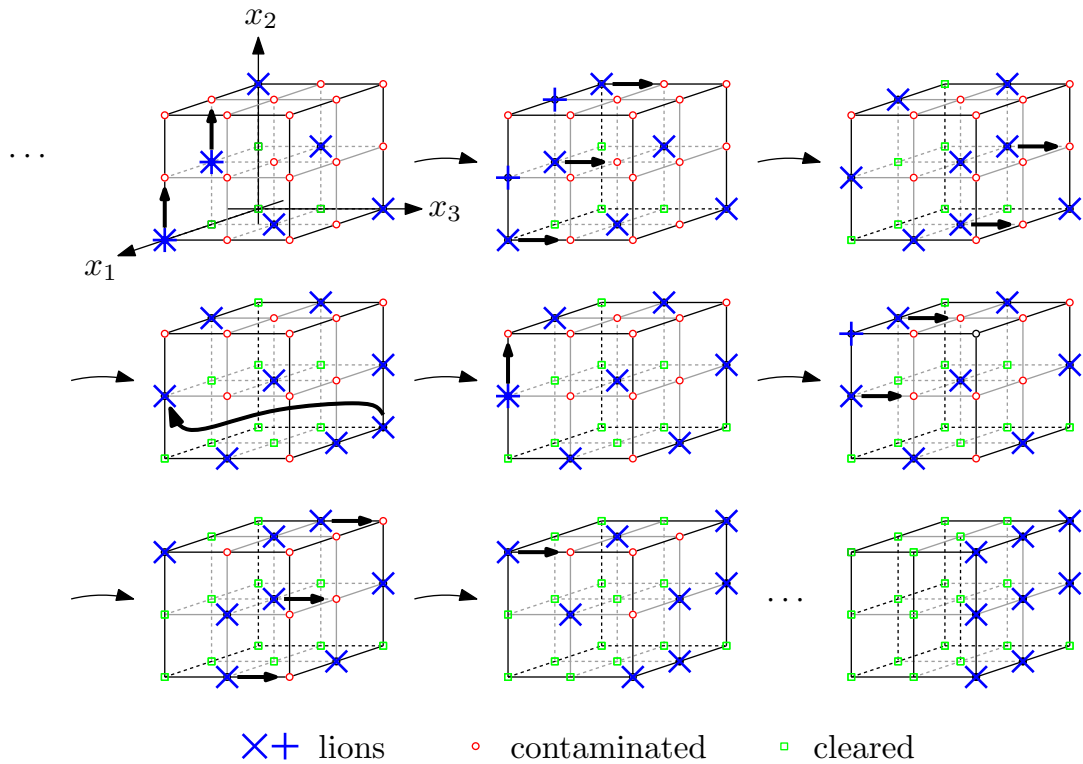


Figure 2.5: Eight lions (rather than $n^{d-1} = 9$) suffice to clear the $3 \times 3 \times 3$ -grid. If two lions occupy the same vertex at the same time, this is indicated by two crosses of different orientation.

The idea is to sweep the vertices layer by layer, where the r -th layer is the set of vertices of L_1 -distance r to the origin. It is denoted by

$$L(r, n, d) := \{v \in [n]^d \mid |v| = r\}.$$

For $r \leq d(n-1)$ the value of $|L(r, n, d)|$ can be computed recursively by:

$$|L(r, n, d)| = \sum_{i=\max(r-n+1, 0)}^{\min(r, (d-1)(n-1))} |L(i, n, d-1)| \quad (2.3)$$

As start of the recursion, we can use $|L(r, n, 1)| = 1$ for $0 \leq r \leq n-1$. To the best of our knowledge, no closed formula for $|L(r, n, d)|$ is known.

Lemma 2.7. $|L(r, n, d)|$ is increasing in r for $0 \leq r \leq \frac{d(n-1)}{2}$.

Proof. We use induction on d . For $d = 1$, we have $|L(r, n, 1)| = 1$ for $0 \leq r \leq \frac{n-1}{2}$, which is monotonously increasing.

Now, let us assume, we already showed for dimension d that $|L(r, n, d)|$ is increasing in r in the range of $0 \leq r \leq \frac{d(n-1)}{2}$. Our task is to prove the monotonicity for dimension $d+1$.

Due to the symmetry $|L(r, n, d)| = |L(d(n-1) - r, n, d)|$, the induction hypothesis implies that $|L(r, n, d)|$ decreases monotonously in r in the range of $\frac{d(n-1)}{2} \leq r \leq d(n-1)$. This symmetry $|L(r, n, d)| = |L(d(n-1) - r, n, d)|$ holds, because all grid points with distance r to the origin have distance $d(n-1) - r$ to the grid point $(n-1, \dots, n-1)$ and vice versa.

Clearly, $|L(r, n, d+1)|$ is monotonously increasing for $0 \leq r \leq n-1$. For the remaining range $n-1 \leq r \leq \frac{(d+1)(n-1)}{2} - 1$, we have $r - n + 1 \geq 0$ and $r + 1 \leq d(n-1)$. Hence, Formula 2.3 leads to

$$\begin{aligned} |L(r+1, n, d+1)| - |L(r, n, d+1)| &= \sum_{i=\max(r-n+2, 0)}^{\min(r+1, d(n-1))} |L(i, n, d)| \\ &\quad - \sum_{i=\max(r-n+1, 0)}^{\min(r, d(n-1))} |L(i, n, d)| \\ &= |L(r+1, n, d)| - |L(r-n+1, n, d)| \\ &=: x(r, n, d) \end{aligned}$$

We will prove the monotonicity of $|L(\cdot, n, d+1)|$ by showing $x(r, n, d) \geq 0$.

If $r+1 \leq \frac{d(n-1)}{2}$, then $r-n+1$ as well as $r+1$ are in the increasing part of $s \mapsto |L(s, n, d)|$, hence $x(r, n, d) \geq 0$ and the monotonicity holds. For $\frac{d(n-1)}{2} < r+1 \leq \frac{(d+1)(n-1)}{2}$, the term $x(r, n, d)$ is monotonously decreasing in r , since $r+1$ is in the decreasing part of $s \mapsto |L(s, n, d)|$, while $r-n+1 \leq \frac{d(n-1)}{2} - \frac{n+1}{2}$ is still in the increasing part. But even if we choose $r = \lfloor \frac{(d+1)(n-1)}{2} \rfloor - 1$, which is the maximum value for r we have to consider, we can still show $x(r, n, d) \geq 0$:

If d is odd or n is odd, we obtain $x(r, n, d) = |L(\frac{d(n-1)}{2} + \frac{n}{2} - \frac{1}{2}, n, d)| - |L(\frac{d(n-1)}{2} - \frac{n}{2} - \frac{1}{2}, n, d)|$, which equals $|L(\frac{d(n-1)}{2} - \frac{n}{2} + \frac{1}{2}, n, d)| - |L(\frac{d(n-1)}{2} - \frac{n}{2} - \frac{1}{2}, n, d)| \geq 0$ due to the symmetry. This shows that the monotonicity of $|L(., n, d)|$ implies the monotonicity of $|L(., n, d+1)|$ within the range needed for Lemma 2.7. Accordingly, if d is even and n is even, we obtain $x(r, n, d) = |L(\frac{d(n-1)}{2} + \frac{n}{2} - 1, n, d)| - |L(\frac{d(n-1)}{2} - \frac{n}{2} - 1, n, d)| \geq 0$. \square

Lemma 2.7 and the symmetry $|L(r, n, d)| = |L(d(n-1) - r, n, d)|$ imply that the *middle layer*, $L(\lfloor \frac{d(n-1)}{2} \rfloor, n, d)$, is the biggest, which will be important for our estimates.

The strategy from Figure 2.5 can be generalized to grid graphs of arbitrary size and dimension. We sweep the grid G_n^d layer after layer. No vertex of $L(r+1, n, d)$ is cleared before the last vertex of $L(r, n, d)$ has been.

Let us consider two consecutive layers $L(r, n, d)$ and $L(r+1, n, d)$, and let us assume that the former has been completely cleared, while the latter is still completely contaminated. Together they form a bipartite graph where every vertex has degree at most d . How many lions are needed to clear $L(r+1, n, d)$ without recontamination?

Lemma 2.8. *Assume that the set $\{v \in [n]^d \mid |v| \leq r\}$ is already cleared. At most $|L(r, n, d)| + |L(r+1, n, d-1)|$ lions are needed to clear $L(r+1, n, d)$ without recontamination, if $r+1 \leq (d-1)(n-1)$. In the remaining case, if $(d-1)(n-1) < r+1 \leq d(n-1)$, we need only $|L(r, n, d)|$ lions.*

Proof. First, we notice that for every $(x_1, \dots, x_d) \in L(r+1, n, d)$ with $x_d \neq 0$, (x_1, \dots, x_{d-1}) lies in $L(r, n, d)$. Furthermore, if there is an edge connecting $(x_1, \dots, x_d) \in L(r+1, n, d)$ and $(y_1, \dots, y_d) \in L(r+1, n, d)$, and $(y_1, \dots, y_d) \neq (x_1, \dots, x_d + 1)$, then $y_d = x_d$.

Now, to clear $L(r+1, n, d)$ we proceed as follows: Initially, we place one lion at each vertex of $L(r, n, d)$. To do that, we obviously need $|L(r, n, d)|$ lions. If $r+1 \leq (d-1)(n-1)$, in addition, for every $y = (y_1, \dots, y_d) \in L(r+1, n, d)$ with $y_d = 0$, we place one lion on a vertex $x = (x_1, \dots, x_d) \in L(r, n, d)$ with $x_d = 0$, so that there exists an edge $(x, y) \in E_n^d$. For that, $|L(r+1, n, d-1)|$ lions are needed. The following algorithm shows that no further lion is necessary to clear $L(r+1, n, d)$.

In the first step each additional lion moves to its assigned vertex $y = (y_1, \dots, y_{d-1}, 0)$. Now all vertices in $L(r+1, n, d)$ of this form have been cleared. Because every vertex of $L(r, n, d)$ is still occupied by a lion, no recontamination has occurred.

Next, we move lions from $L(r, n, d)$ to $L(r+1, n, d)$ without causing any recontamination. Beginning with $i = 0$, we repeat the following action until i equals $n-1$ or r : Move every lion that is placed on a vertex of the form (x_1, \dots, x_{d-1}, i) to $(x_1, \dots, x_{d-1}, i+1)$.

After this loop, every vertex of $L(r+1, n, d)$ is occupied, as follows from the remarks at the beginning of this proof. Moreover, no recontamination was possible during the loop, because every lion left his vertex via the only edge that led a contaminated

vertex.

Note that in the case of $r + 1 > (d - 1)(n - 1)$ the layer $L(r + 1, n, d)$ does not contain any vertex v with $v_d = 0$. We can do the same as above with $|L(r, n, d)|$ lions. No additional lions are needed. \square

As a consequence, we obtain the following upper bounds.

Lemma 2.9.

$$\begin{aligned} k_d(n) &\leq |L(\lfloor \frac{d(n-1)}{2} \rfloor, n, d)| + |L(\lfloor \frac{d(n-1)}{2} \rfloor + 1, n, d - 1)| \\ &\leq 2|L(\lfloor \frac{d(n-1)}{2} \rfloor, n, d)| \end{aligned}$$

Proof. From Lemma 2.8 we can conclude

$$\begin{aligned} k_d(n) &\leq \max \left(\begin{array}{c} \max_{0 \leq r < (d-1)(n-1)} |L(r, n, d)| + |L(r + 1, n, d - 1)| \\ \max_{(d-1)(n-1) \leq r < d(n-1)} |L(r, n, d)| \end{array} \right) \\ &\leq \max_{0 \leq r < d(n-1)} |L(r, n, d)| + |L(r + 1, n, d - 1)|. \end{aligned}$$

For $r + 1 \leq (d - 1)(n - 1)$, the recursive formula for $|L(r, n, d)|$ implies

$$|L(r, n, d)| + |L(r + 1, n, d - 1)| = \sum_{i=\max(0, r-n+1)}^{r+1} |L(i, n, d - 1)|.$$

We will establish that $\sum_{i=\max(0, r-n+1)}^{r+1} |L(i, n, d - 1)|$ obtains its maximum for $r = \lfloor \frac{d(n-1)}{2} \rfloor$. The above sum contains at most $n + 1$ consecutive elements of the sequence:

$$(|L(i, n, d - 1)|)_{0 \leq i \leq (d-1)(n-1)}$$

By Lemma 2.7 and its proof, we know that this sequence is monotonously increasing for $i \leq \frac{(d-1)(n-1)}{2}$, monotonously decreasing for $i \geq \frac{(d-1)(n-1)}{2}$ and symmetric with respect to $i = \frac{(d-1)(n-1)}{2}$. Hence, the sum is maximized if $n + 1$ consecutive sequence indices are chosen as symmetrically as possible around $\frac{(d-1)(n-1)}{2}$.

If $n + 1$ is odd and d is odd, a maximum sum arises if $r + 1 - \frac{n}{2} = \frac{(d-1)(n-1)}{2}$. If $n + 1$ is odd and d is even, a maximum sum arises if $r + 1 - \frac{n}{2} = \frac{(d-1)(n-1)}{2} + \frac{1}{2}$. In the third case, if $n + 1$ is even, a maximum sum is obtained for $r - \frac{n}{2} + \frac{1}{2} = \frac{(d-1)(n-1)}{2}$.

In all three cases, the sum $\sum_{i=\max(0, r-n+1)}^{r+1} |L(i, n, d - 1)|$ obtains its maximum for $r = \lfloor \frac{d(n-1)}{2} \rfloor$. This proves the first inequality of the Lemma. The second one is a simple but not very tight estimate. \square

2.4.2 Lower bound

Now, that we found an upper bound on $k_d(n)$ in terms of the size of the middle layer, we want to prove that $L := |L(\lfloor \frac{d(n-1)}{2} \rfloor, n, d)|$ also determines a lower bound.

The proof is based on a result by Bollabás and Leader [20] which uses the following *simplicial order* $<$ on $[n]^d$. For every two vertices $v, w \in [n]^d$ we have

$$v < w \quad :\Leftrightarrow \quad |v| < |w| \vee \\ (|v| = |w| \wedge \exists j \in [n] : (v_j > w_j \wedge \forall i < j : v_i = w_i))$$

Let $v^1 := 0 := (0, \dots, 0)$, $v^2 := (1, 0, \dots, 0)$, $v^3 := (0, 1, 0, \dots, 0)$, ... denote the vertices in simplicial order. And for every $i \in \{1, \dots, n^d\}$ let $A_i := \{v^1, \dots, v^i\}$ be the set of the first i vertices with respect to the simplicial order. Then, $|\partial A_i|$ denotes the number of boundary vertices of A_i .

With this notation we can formulate the following important kind of isoperimetric inequality.

Lemma 2.10. *For every given size $m \in \{1, \dots, n^d\}$ the set A_m attains the minimum number of boundary vertices, i.e.*

$$\forall C \subseteq [n]^d : |C| = m \Rightarrow |\partial C| \geq |\partial A_m|.$$

Proof. Remember that $\mathcal{N}(A)$ denotes the closed neighborhood of A which is defined as $\mathcal{N}(A) = \{v \in V \mid \exists w \in A : |v - w| \leq 1\}$. Theorem 8 in [20] states that every $C \subseteq [n]^d$ of $m := |C|$ vertices satisfies $|\mathcal{N}(C)| \geq |\mathcal{N}(A_m)|$. This can be translated into our Lemma 2.10 as follows:

$$|\partial C| = |\mathcal{N}(\overline{C})| - |\overline{C}| = |\mathcal{N}(\overline{C})| - |\overline{A_m}| \geq |\mathcal{N}(\overline{A_m})| - |\overline{A_m}| = |\partial A_m|$$

□

Now we can prove a lower bound in terms of the middle layer.

Lemma 2.11. $k_d(n) \geq \lfloor \frac{1}{6}L \rfloor$.

Proof. Let us consider the paths of $k_d(n)$ lions successfully clearing the grid. By Lemma 2.2, the number of cleared vertices increases by at most $k_d(n)$ within each step. Consequently, there must be some moment t such that $\frac{n^d}{2} - k_d(n) \leq |\mathcal{C}(t)| \leq \frac{n^d}{2}$ and $|\mathcal{C}(t+1)| > |\mathcal{C}(t)|$. Thanks to Lemma 2.3, we obtain

$$k_d(n) \geq \frac{1}{2}|\partial \mathcal{C}(t)|.$$

Thus we have to lower bound the number of boundary vertices of a subset of the grid with size $|\mathcal{C}(t)|$. By Lemma 2.10, the fewest boundary vertices for such a set are obtained by the set $A_{|\mathcal{C}(t)|}$. Thus, $k_d(n) \geq \frac{1}{2}|\partial A_{|\mathcal{C}(t)|}|$.

Let us assume there exist n, d such that $k_d(n) < \lfloor \frac{1}{6}L \rfloor$. Then both nodes $v^{\lfloor \frac{n^d}{2} \rfloor - k_d(n)}$ and $v^{\lfloor \frac{n^d}{2} \rfloor}$ are contained in the middle layer, $L(\lfloor \frac{d(n-1)}{2} \rfloor, n, d)$. We want to estimate the number of boundary vertices of $A_{|\mathcal{C}(t)|}$ in the middle layer. Let us first consider the situation for the set $A_{\lfloor \frac{n^d}{2} \rfloor}$, which contains half the number of all grid vertices. If $d(n-1)$ is even, half of the vertices of the middle layer are contained in $A_{\lfloor \frac{n^d}{2} \rfloor}$. If $d(n-1)$ is odd, $A_{\lfloor \frac{n^d}{2} \rfloor}$ contains all vertices of the middle layer and no vertex of the neighboring layer with distance $\lceil \frac{d(n-1)}{2} \rceil$ to the origin. In both cases, there are at least $\lfloor \frac{1}{2}L \rfloor$ boundary vertices of $A_{\lfloor \frac{n^d}{2} \rfloor}$ in the middle layer of the grid.

Since $\frac{n^d}{2} - k_d(n) \leq |\mathcal{C}(t)| \leq \frac{n^d}{2}$ and $k_d(n) < \lfloor \frac{L}{6} \rfloor$, the sets $A_{\lfloor \frac{n^d}{2} \rfloor}$ and $A_{|\mathcal{C}(t)|}$ differ in at most $k_d(n)$ middle layer vertices. We conclude that $A_{|\mathcal{C}(t)|}$ contains at least $\lfloor \frac{1}{2}L \rfloor - k_d(n)$ boundary vertices in the middle layer. Hence,

$$k_d(n) \geq \frac{1}{2} |\partial A_{|\mathcal{C}(t)|}| \geq \frac{1}{2} \left\lfloor \frac{L}{2} \right\rfloor - \frac{1}{2} k_d(n) > \frac{1}{2} \left\lfloor \frac{L}{2} \right\rfloor - \frac{1}{2} \left\lfloor \frac{L}{6} \right\rfloor \geq \left\lfloor \frac{L}{6} \right\rfloor$$

To verify the last inequality, we notice that the value of $\frac{1}{2} \lfloor \frac{L}{2} \rfloor - \frac{3}{2} \lfloor \frac{L}{6} \rfloor$ depends only on $L \bmod 6$, thus it is sufficient to check the cases $0 \leq L \leq 5$. What we obtained is a contradiction to our assumption $k_d(n) < \lfloor \frac{L}{6} \rfloor$. \square

2.4.3 Asymptotic estimate

We have given upper and lower bounds on $k_d(n)$ in terms of the size of the middle layer, thereby proving $k_d(n) \in \Theta(L(\lfloor \frac{d(n-1)}{2} \rfloor, n, d))$. But how does this size grow? Unfortunately, we are not aware of a closed formula for $|L(\lfloor \frac{d(n-1)}{2} \rfloor, n, d)|$ in general. But the following folklore result [74, sequence number A077042] describes an asymptotic estimate.

Lemma 2.12. $|L(\lfloor \frac{d(n-1)}{2} \rfloor, n, d)| \in \Theta(\frac{n^{d-1}}{\sqrt{d}})$

Proof. Let us consider d independent and identically distributed random variables X_1, \dots, X_d each of which can take any integer value from 0 to $n-1$ with equal probability. Each X_i has expectation $\frac{n-1}{2}$ and variance $\frac{n^2-1}{12}$. Therefore, the random variable $S_d^n := X_1 + \dots + X_d$ has expectation $\frac{(n-1)d}{2}$ and variance $\frac{d(n^2-1)}{12}$. By the central limit theorem, the distribution of $\frac{S_d^n - \frac{(n-1)d}{2}}{\sqrt{\frac{d(n^2-1)}{12}}}$ converges towards the standard normal distribution. This means, that the probability of the event $S_d^n = \lfloor \frac{d(n-1)}{2} \rfloor$ asymptotically behaves like $\frac{1}{\sqrt{2\pi}} \int_{-1/(2\sqrt{d(n^2-1)/12})}^{1/(2\sqrt{d(n^2-1)/12})} \exp(-\frac{t^2}{2}) dt$. For increasing n and/or d , the integration domain shrinks around 0, hence the integrand stays in an interval $[1-\varepsilon, 1]$

where $\varepsilon \searrow 0$. The probability lies in the interval $\left[(1 - \varepsilon) \sqrt{\frac{6}{\pi d(n^2 - 1)}}, \sqrt{\frac{6}{\pi d(n^2 - 1)}} \right]$. Multiplying this by n^d , which is the number of all vertices in the grid, leads to the fact that the number of vertices in the middle layer asymptotically behaves like $\Theta\left(\frac{n^{d-1}}{\sqrt{d}}\right)$. \square

We can combine Lemmas 2.9, 2.11 and 2.12 to

Theorem 2.13. $k_d(n) \in \Theta(|L(\lfloor \frac{d(n-1)}{2} \rfloor, n, d)|) = \Theta\left(\frac{n^{d-1}}{\sqrt{d}}\right)$.

2.5 Flying lions in the 2-dimensional grid

We are not aware of any strategy with fewer than n lions to clear the two-dimensional $n \times n$ -grid. Indeed, we conjecture that n is the clearing number of the $n \times n$ -grid. In this section, we will establish that fewer than n lions are sufficient, if the lions are able to fly.

To be more precise, we consider the problem variant where the movement of the lions is not restricted to the edges of the grid. If a lion moves from a grid vertex to an adjacent grid vertex, this still prevents the contamination from using the corresponding edge in the other direction. However, if a lion flies from a grid vertex to another vertex which is not adjacent in the grid, no edge is protected.

Our proof of Theorem 2.1 in Section 2.3 does not take any advantage of the fact that the movement of the lions is restricted to grid edges. This means that the lower bound of $\frac{n}{2}$ is still valid for the case of flying lions.

Our main result is that $\frac{n}{2} + O(\sqrt{n})$ flying lions can clear the $n \times n$ -grid. However, we start with showing the weaker result that $\frac{2}{3}n + O(1)$ lions are enough in order to illustrate the idea of our method in a simpler case.

For $h := 2\lceil \frac{n}{3} \rceil$, we denote the grid vertices with x -coordinate at most $h - 1$ as the *left part of the grid*, while the vertices with x -coordinate at least $h + 1$ are the *right part*. The vertices with x -coordinate equal to h are called *separator vertices*. Our strategy guarantees that the set of lions occupying separator vertices stays the same all the time. Lions belonging to this set are called *separator lions*.

We will establish a strategy enabling $h + 11$ lions to clear one row of the grid after the other.

Suppose, we are given a situation where the 0th, 1st, \dots , v -th row of the grid are free of contamination, where $5 \leq v \leq n - 6$. The vertices $(0, v), (1, v), \dots, (h - 1, v)$ as well as the separator vertices $(h, v - 5), (h, v - 4), \dots, (h, v + 5)$ are occupied by the lions.

The movement of the lions is illustrated in Figure 2.6. The 11 lions located on the separator vertices keep their position until all of them move one row upwards when

Notice that, if $v < 5$ or $v > n - 6$, the situation is even better for the lions. Starting from the 0th row, they can consecutively clear one row after the other. After each gained row, all separator lions move one row upwards during the next time step. This happens simultaneously to the lions in the left part of the grid moving upwards. The total number of lions needed for this strategy is $h + 11 \in \frac{2}{3}n + O(1)$.

Lemma 2.14. $\frac{2}{3}n + O(1)$ flying lions can clear the $n \times n$ -grid.

For a strategy in the above manner, the speed ratio between the upwards moving lions in one part of the grid and the downwards moving contamination in the other part does not have to be 2. If we save some lions, the speed advantage of the lions compared with the contamination gets smaller than 2, but the algorithm can still be performed. However, if we save too many lions, the problem is that the speed advantage gets so small that the number of separator lions dominates. It will turn out that a total number of $\frac{n}{2} + \Theta(\sqrt{n})$ lions results in the best tradeoff between the number of lions operating in one part of the grid and the number of separator lions.

How does the speed advantage depend on the number of lions? Let us consider a grid with s columns and sufficiently many rows. As Figure 2.7 shows, $s + 2$ lions can gain s rows in $s - 1$ steps.

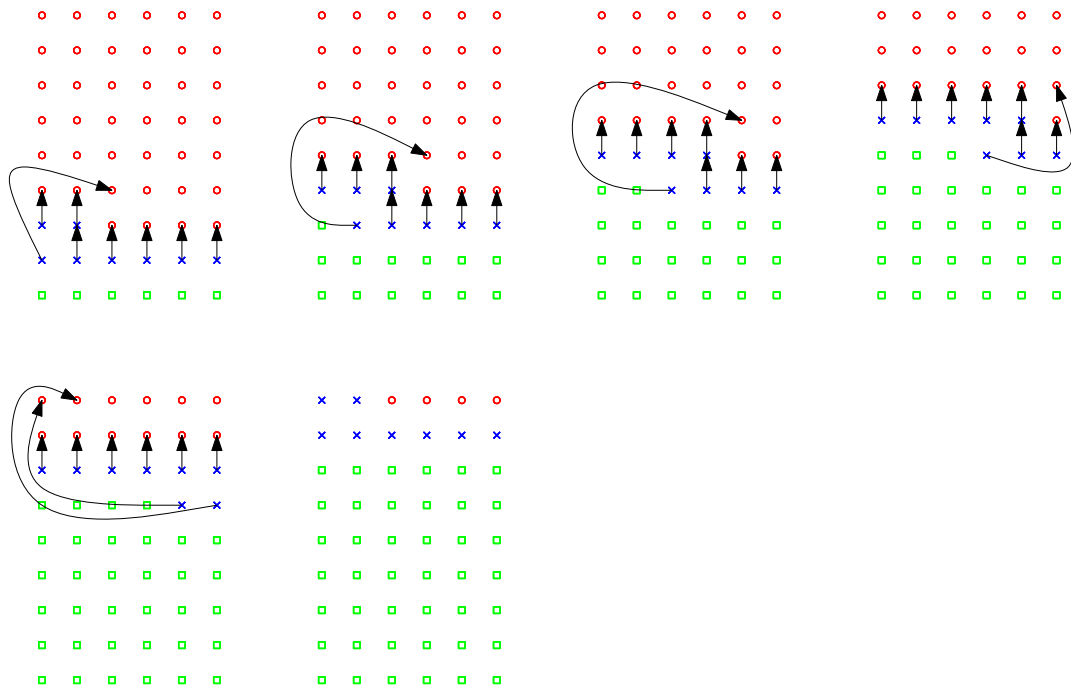


Figure 2.7: Speed advantage for the lions: they gain 6 rows in 5 steps.

This observation can be generalized to the following Lemma.

Lemma 2.15. *Suppose, we are given a grid with s columns and sufficiently many rows. For $2 \leq r < s$, there is a strategy enabling $s + r$ lions to gain $\lceil \frac{s-r}{r-1} \rceil + 2$ rows in $\lceil \frac{s-r}{r-1} \rceil + 1$ steps.*

Proof. The statement is restricted to a grid with s columns. No contamination from outside of these s columns has to be taken into account. The lions are using the strategy from Figure 2.7. We assume their start locations to be the s vertices of the v -th row and the r leftmost vertices of the $(v + 1)$ -th row. Our strategy ensures that, at any time, there are only 2 rows containing lions. These rows are called *active rows*. Let w be the rightmost vertex occupied by a lion in the upper active row. All lions in the upper active row move one row upwards. Also, all lions in the lower active row which are right of w or directly under w move one row upwards. We still have $r - 1$ lions left. These lions fly to vertices two rows upwards such that no gaps in this target row arises.

We repeat such moves until the upper active row is completely filled up with lions. Since each step increases the number of cleared vertices in the upper active row by $r - 1$, exactly $\lceil \frac{s-r}{r-1} \rceil$ steps are needed. Afterwards, one more step is used to move all lions in the upper active row upwards, while the other lions fly three rows upwards to the leftmost positions. \square

Now, we are ready to prove that $\frac{n}{2} + O(\sqrt{n})$ flying lions are sufficient to clear the grid. This time, we choose the *left* and the *right part* of the grid of equal size.

The initial situation we consider is that the grid is cleared from the 0-th row up to the v -th row. We have $\frac{n}{2}$ lions located in the complete left part of the v -th row and $\lceil \sqrt{n} \rceil$ lions on the leftmost vertices of the $(v + 1)$ -th row. In terms of Lemma 2.15, we have $s = \frac{n}{2}$ and $r = \lceil \sqrt{n} \rceil$. For these values of s and r , it can be easily verified that $\lceil \frac{s-r}{r-1} \rceil + 1 \leq \lceil \sqrt{n} \rceil$ for all $n \geq 2$.

Hence, by Lemma 2.15, we conclude that, if the lions in the left part of the grid move upwards for $3\lceil \sqrt{n} \rceil$ steps, they can gain $3\lceil \sqrt{n} \rceil + 3$ rows there. During this time, the cleared area in the right part of the grid shrinks only by $3\lceil \sqrt{n} \rceil$ rows.

Now, the lions fly to their appropriate positions in the right part of the grid. Afterwards, $3\lceil \sqrt{n} \rceil + 2$ rows remain gained in the left part—compared with the initial situation. In the right part, the lions occupy the complete $(v - 3\lceil \sqrt{n} \rceil - 1)$ -th row and the $\lceil \sqrt{n} \rceil$ leftmost vertices of the $(v - 3\lceil \sqrt{n} \rceil)$ -th row.

In the following, the lions in the right part of the grid move upwards for $3\lceil \sqrt{n} \rceil$ steps. Comparing the situation afterwards with the initial situation, 2 rows are gained in both parts of the grid. Then, if the lions fly from right to left again, we have the initial situation with $v + 1$ instead of v . Or, to say it with other words, the lions gained one row.

Besides the $\frac{n}{2} + \lceil \sqrt{n} \rceil$ lions operating either on the left or the right part of the grid, we also need the separator lions. The maximum vertical boundary they have to protect occurs when the lions are flying from left to right. A number of $6\lceil \sqrt{n} \rceil + 3$ separator lions is sufficient. This means that the total number of all lions involved in this strategy is $\frac{n}{2} + 7\lceil \sqrt{n} \rceil + 3$.

Finally, we obtain:

Theorem 2.16. *In the case of flying lions in the 2-dimensional grid, the clearing number is at least $\frac{n}{2}$ and at most $\frac{n}{2} + O(\sqrt{n})$.*

2.6 Conclusion

We conclude that the number of flying lions needed to clear the $n \times n$ grid is between $\frac{n}{2}$ and $\frac{n}{2} + O(\sqrt{n})$. Which better lower or upper bounds can be proved? Another interesting question is whether this clearing strategy can be generalized to higher dimensions.

If the lions' movement is restricted to the edges of the grid, the two-dimensional clearing number is between $\frac{n}{2}$ and n . Unfortunately, we could not reduce this gap although we spent much effort in trying to do so. Without any strong evidence, it feels to us that the search number in this case is indeed n and that concepts from graph theory (like path-width or tree-width) might help to show new results.

Chapter 3

Meeting Scheduling respecting Time and Space

3.1 Introduction

We consider the following problem: a set of people, called participants, would like to schedule a meeting. The participants are located at different sites and would like to find a common point (not necessarily one of the sites) for the meeting to be held. These sites are rather far from each other, so that the travel times to the meeting point are important. We assume that the travel distances between locations can be measured using the Euclidean distance in the plane.

Participants have individual schedules which specify: the earliest possible time they can leave their current location and the latest possible time they must arrive at their next location. We wish to solve the problem of finding a meeting point so that the time for all participants to meet is maximized.

Our general objective for this research is to derive efficient algorithms for solving general meeting scheduling problems, to find approximate solutions, where appropriate, to implement our solutions, and to integrate them into applications that allow users who are connected over a network to schedule meetings. Most current meeting scheduling systems take into consideration only time and not location/geometry, see *e.g.* [11, 39, 71, 72]. The systems in [25, 40] apply distributed meeting scheduling algorithms. Rasmussen and Trick [66] define the timetable constrained distance minimization problem, which is a sports scheduling problem applicable for tournaments where the total travel distance must be minimized. They use integer programming and constraint programming formulation for the problem. Integer programming is also applied by Wang *et al.* [75] for scheduling meetings among students and teachers in universities. For another scheduling approach to school meetings (between teachers and parents), Rinaldi *et al.* [68] set weights and build directed graphs based on the time slots, then they search shortest paths in the resulting graphs.

Some meeting scheduling systems allow as input a list of possible meeting locations and then select, from that list, one location based on different optimization criteria (such as time to meeting or cost of meeting). Examples of such work include: [5, 69, 34]. The algorithms used in these papers are relatively simple. They repeatedly execute a minimum cost algorithm for each of the preselected locations and then select the best option. Here, we are dealing with the more complex problem instance, where we are not given preselected locations, but rather have to compute an optimum location which could be anywhere on the plane.

Chapter 3 is structured as follows.

In Section 3.2 we introduce the formal problem definition. The meeting problem is converted to a computational geometry one in Section 3.3. In Section 3.4, we establish that the corresponding computational geometry problem can be regarded as an LP-type problem whose combinatorial dimension equals 4. This observation leads to a randomized algorithm with expected running time $O(n)$ improving the previously known $O(n \log n)$ result by [43]. We published these results first in [16, 17].

The following sections deal with some variants of the original problem. The problem to determine a meeting of maximum duration for all except k participants is analyzed in Section 3.5. Afterwards, in Section 3.6, we consider the problem to maximize the number of participants for a common meeting. An NP-hardness proof for the problem variant where n participants shall meet as long as possible at k different locations is given in Section 3.7. Finally, in Section 3.8, we consider the problem variant where the number of previously scheduled meetings for each participant may be greater than 2.

3.2 Problem definition

We assume a situation where all participants can travel at the same constant speed v in the plane. Since the precise value of v is not critical, we may as well assume that $v = 1$ holds. Then, the time it takes to travel from location l to location l' is given by the Euclidean distance $\|l' - l\|$.

Let $M := \{M_1, \dots, M_n\}$ denote the set of participants. The i th participant M_i has a previous meeting at location $l_i^{pre} \in \mathbb{R}^2$ that lasts until time $t_i^{pre} \in \mathbb{R}$. He is due for a subsequent meeting that starts at time $t_i^{sub} \in \mathbb{R}$ at location $l_i^{sub} \in \mathbb{R}^2$.

We want to schedule a meeting of longest possible duration in between the previous and subsequent meetings of all participants (or report that no such meeting is possible).

To solve our problem, we have to find a location $x \in \mathbb{R}^2$ such that the time interval between the arrival of the last participants from their previous meetings at time

$$S(x) := \max_{1 \leq i \leq n} \left(t_i^{pre} + \|l_i^{pre} - x\| \right)$$

and the departure of the first participants to their subsequent meetings at time

$$E(x) := \min_{1 \leq i \leq n} \left(t_i^{sub} - \|l_i^{sub} - x\| \right)$$

is maximized.

We define $f(x) := E(x) - S(x)$ and are interested in finding a location x for which $f(x)$ is maximum. Of course, no common meeting is possible if the maximum value of f is negative. If we add the same number c to all parameters t_i^{sub} , then the value of $f(x)$ increases by c for any location x , but the maximum location does not change. Hence, we can assume that there is a solution x with $f(x) > 0$ in the following.

As an example for the meeting scheduling problem, we consider the following instance, where every participant has to return to the point of his previous meeting. To be more precise, we have $\forall i : 1 \leq i \leq n$:

- $l_i^{pre} = l_i^{sub}$,
- $t_i^{pre} = 0$, and
- $t_i^{sub} = 1 + 2 \max_{2 \leq j \leq n} \|l_1^{pre} - l_j^{pre}\| =: C$.

The way we have chosen C ensures that $f(l_1^{pre}) = 1$, hence there are feasible locations for the new meeting. The best one can do for this instance is to find a location where the participants can arrive earliest, because of the symmetry this is also a location where they can stay longest.

Every participant can reach the location x at time $S(x)$. This means that all l_i^{pre} are within a disk with midpoint x and radius $S(x)$. The center of the *smallest enclosing disk* of the l_i^{pre} corresponds to the optimum solution.

There is a randomized algorithm for computing the smallest enclosing disk of n points in the plane with expected running time $O(n)$, see Welzl [76]. LP-type problems, introduced by Sharir and Welzl [73], build an abstract and more general framework for Welzl's algorithm for the smallest enclosing disk. We will use this concept of LP-type problems for our solution in Section 3.4.

3.3 Geometric interpretation of the problem

The possible locations participant M_i can reach, once his previous meeting has ended, are given by a system of circles centered at l_i^{pre} that start expanding at

time t_i^{pre} . If we use the time as a third coordinate, the expanding circles for participant M_i form a vertical cone PRE_i with apex angle $\frac{\pi}{2}$ whose bottommost point is its apex $((l_i^{pre})_X, (l_i^{pre})_Y, t_i^{pre})$. Let SUB_i denote the corresponding cone for the subsequent meeting, so SUB_i is a vertical cone whose uppermost point is its apex $((l_i^{sub})_X, (l_i^{sub})_Y, t_i^{sub})$.

We define $PRE := \bigcap_{i=1}^n PRE_i$, $SUB := \bigcap_{i=1}^n SUB_i$ and $F := PRE \cap SUB$.

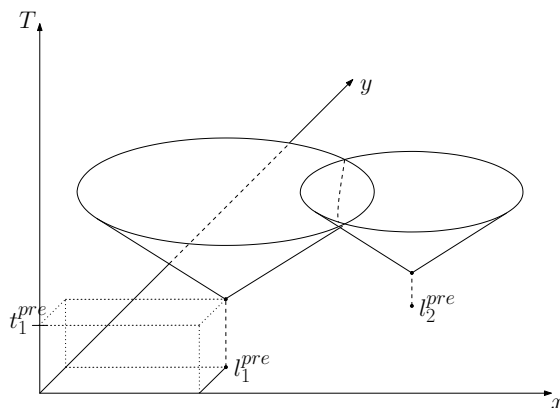


Figure 3.1: PRE is the intersection of upward cones with apex angle $\frac{\pi}{2}$

We call F the set of *feasible points* because $(l_X, l_Y, t) \in F$ means that all participants can meet at location l at time t . Observe that F is convex as the intersection of convex sets. A meeting at location l is possible from time t_1 to t_2 , where $t_1 \leq t_2$, if and only if the vertical line segment with endpoints (l_X, l_Y, t_1) and (l_X, l_Y, t_2) is fully contained in F . Consequently, we have to *find a vertical line segment of maximum length contained in F* .

In [43], the authors present a sweep algorithm to compute such a segment. Their algorithm uses the fact that the projection of the cones in PRE on the XY -plane is a *furthest site Voronoi diagram of circles*, see [65, 55], and leads to the following result.

Theorem 3.1. *One can compute a meeting of longest duration (or determine that no meeting is possible) for a set of n participants in time $O(n \log n)$.*

Proof. The proof can be found in [43, 16, 17]. □

3.4 LP-approach

In this section, we provide a solution to the meeting scheduling problem using the concept of LP-type problems. We will show that the problem can be stated as an LP-type problem and that its dimension equals 4. This enables us to compute a meeting of longest duration for n participants in expected time $O(n)$.

3.4.1 General framework of LP-type problems

LP-type problems were introduced by Sharir and Welzl [73]. Here we will follow the slightly modified description by Matoušek [53]. For convenience, we describe the problem as a maximization problem.

A *maximization problem* is a pair (H, w) , where H is a finite set, and $w : 2^H \rightarrow W$ is a function with values in a linearly ordered set (W, \leq) . The elements of H are called the *constraints*, and for a subset $G \subset H$, $w(G)$ is called the *value* of G . Intuitively, the value $w(G)$ for a subset G of constraints stands for the largest value attainable by a certain objective function while satisfying all the constraints of G . The goal is to find $w(H)$. The set W is assumed to possess a largest element denoted by ∞ (intuitively, it stands for ‘optimum undefined’) and usually also a smallest element $-\infty$ (with intuitive meaning ‘no feasible solution exists’).

The maximization problem (H, w) is called an *LP-type problem* if the following two axioms are satisfied:

Axiom 1. (Monotonicity) For any F, G with $F \subset G \subset H$, $w(F) \geq w(G)$.

Axiom 2. (Locality) For any $F \subset G \subset H$ with $w(F) = w(G) < \infty$ and any $h \in H$, $w(G \cup \{h\}) < w(G)$ implies that also $w(F \cup \{h\}) < w(F)$.

We recall some terminology from the theory of LP-type problems. A *basis* B is a set of constraints with $w(B') > w(B)$ for all proper subsets B' of B . A *basis for a subset* G of H is a basis $B \subset G$ with $w(B) = w(G)$. The maximum cardinality of any *basis* is called the *dimension* of (H, w) and denoted by $\dim(H, w)$. We say that a constraint $h \in H$ violates a set G if $w(G \cup \{h\}) < w(G)$.

There are several algorithms for solving an LP-type problem of constant bounded dimension in time $O(|H|)$. The algorithms differ in the assumptions on the primitive operations available for the considered problem. The primitive operations of Sharir and Welzl are:

Violation test. Given a basis B and a constraint $h \in H$, decide whether h violates B .

Basis change. Given a basis B and a constraint h , return (some) basis for $B \cup \{h\}$.

Initial basis. In the beginning, we have some basis B_0 with $w(B_0) < \infty$.

The expected number of primitive operations performed by the algorithm by Sharir and Welzl is $O(|H|)$ for any LP-type problem (H, w) with constant bounded dimension.

3.4.2 Preparations for applying the general framework

Is the meeting scheduling problem, as we stated it until now, an LP-type problem, if we consider the set of participants M as the set of constraints and let the function w map a subset $M' \subset M$ to the optimum meeting duration for all participants in M' ? The answer to this question is negative, because Axiom 2 does not hold in general.

Figure 3.2 shows a situation where Axiom 2 is not fulfilled.

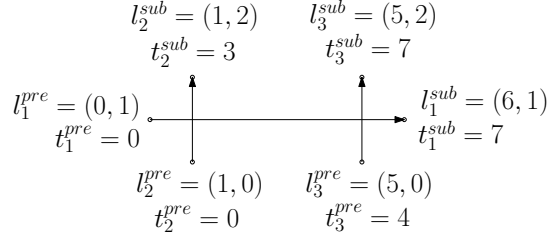


Figure 3.2: Axiom 2 does not hold for the participants M_1, M_2, M_3 , if we use $x \mapsto f(x)$ as objective function. For the sets $F := \{M_1\}$, $G := \{M_1, M_2\}$ and $h := M_3$, we obtain $w(F) = w(G) = 1$ and $w(G \cup \{h\}) = -\infty < w(G)$, but $1 = w(F \cup \{h\}) = w(F)$.

We will establish that the problem becomes an LP-type problem if we lexicographically maximize (f, S) instead of just maximizing f , where $S(x)$ again denotes the time of the latest arrival of the participants at location x . If we use (f, S) as objective function in the example of Figure 3.2, Axiom 2 is no longer violated with respect to the sets $F, G, G \cup \{h\}$, because the solution of F which is $(1, 0)$ is better than the solution of G which is $(1, 1)$.

As Figure 3.2 indicates, we have to analyze those situations where f obtains its maximum in more than one point.

Lemma 3.2. $|S(x_1) - S(x_2)| = \|x_1 - x_2\|$ holds, if both locations x_1, x_2 maximize f .

Proof. Since the set of feasible points F is convex, the line segment B_S from $(x_1, S(x_1))$ to $(x_2, S(x_2))$ is completely contained in F . Analogously, the line segment B_E from $(x_1, E(x_1))$ to $(x_2, E(x_2))$ is in F . The situation is shown in Figure 3.3.

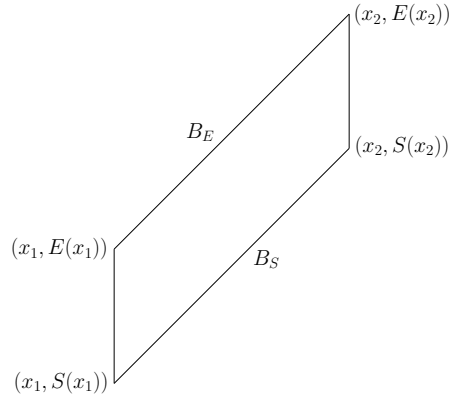


Figure 3.3: Every vertical segment of maximum length contained in this parallelogram corresponds to a meeting of longest duration.

If $(l, z) \in B_S$ were an inner point of F , we could schedule a meeting at l of longer duration than at x_1 . But this would be a contradiction to the assumption that x_1 maximizes f . Hence, B_S belongs to the boundary ∂F of F . There must be some cone PRE_j whose boundary completely contains B_S . Otherwise, the line segment B_S could intersect the boundary of every cone in at most 2 points. Finally, $B_S \subset \partial PRE_j$ implies that B_S is a line segment of slope 1, which proves the Lemma. \square

Lemma 3.3. *There is a unique location x lexicographically maximizing $(f(x), S(x))$.*

Proof. Consider the disk $C := \{x \in \mathbb{R}^2 : \|l_1^{pre} - x\| \leq t_1^{sub} - t_1^{pre} - f(l_1^{pre})\}$ centered at l_1^{pre} . Then $f(x) < f(l_1^{pre})$ holds for all $x \in \mathbb{R}^2 \setminus C$. Due to the compactness of C and the continuity of $f|_C$, the function $f|_C$ obtains its maximum T on C . Since $T \geq f(l_1^{pre}) > f(x)$ for all $x \in \mathbb{R}^2 \setminus C$, T is also the maximum of f on \mathbb{R}^2 . The continuous function $S|_{f^{-1}(T)}$ obtains its maximum t on the compact set $f^{-1}(T)$. Thanks to Lemma 3.2, there can be at most one point x , such that $f(x) = T$ and $S(x) = t$. Hence, the optimum point is unique. \square

Lemma 3.4. *The maximum meeting duration is already determined by at most 2 participants, if f obtains its maximum in more than one point.*

Proof. Let x_1 and x_2 denote distinct points where f obtains its maximum and let B_S, B_E and PRE_j denote the same things as in the proof of Lemma 3.2. An analogous argument to the one showing that $B_S \subset \partial PRE_j$ leads to the fact that there is a downward cone SUB_k such that $B_E \subset \partial SUB_k$. The optimum meeting duration T is obtained for a piece of the line segment $l_j^{pre} l_k^{sub}$. But even a 2-participant-meeting just of M_j and M_k cannot have a duration longer than T . If $j = k$, which is also possible, the maximum meeting duration is already determined by only one participant. \square

Lemma 3.5. *The locations maximizing f form a line segment.*

Proof. If there is only one point where f obtains its maximum T , we are done. Otherwise, let x_1, x_2 and M_j, M_k denote the same things as before. As we have seen in the proof of Lemma 3.2, f obtains its maximum T for all points on the line segment from x_1 to x_2 . On the other hand, every point not located on the line through $l_j^{pre} l_k^{sub}$ already prevents only the participants M_j, M_k to have a meeting of duration T . Hence, the points maximizing f form a line segment. \square

Now, we introduce some terminology. For $M' \subset M$ we refer to $OPT_{M'}$ as the lexicographically maximum pair $(f(x), S(x))$ for a meeting of all participants in M' . Let $f|_{M'}(x)$ denote the maximum duration for a meeting at location x of all participants in M' . Let $L(OPT_{M'})$ denote the unique location where this optimum meeting is held.

3.4.3 Application of the general framework

We consider the set of participants M as the set of constraints. The function w maps a subset $M' \subset M$ to the pair $w(M') := OPT_{M'}$. These pairs are compared lexicographically.

Theorem 3.6. *To find the optimum meeting for a set of participants is an LP-type problem.*

Proof. Clearly, a meeting, which is possible for a set of participants G , is also possible for every subset of G . Thus Axiom 1 holds.

To show that Axiom 2 holds, we consider sets $F \subset G \subset M$ and a participant $h \in M$ such that $OPT_F = OPT_G$ and $OPT_{G \cup \{h\}} < OPT_G$. According to Lemma 3.3, every point different from $L(OPT_F)$ leads to a solution worse than $OPT_F = OPT_G$ already for the participants in F . Hence, every point different from $L(OPT_F)$ leads also for the participants in G to a solution worse than $OPT_F = OPT_G$. We conclude $L(OPT_F) = L(OPT_G)$. The inequality $OPT_{G \cup \{h\}} < OPT_G$ means that participant h cannot stay at the location $L(OPT_G)$ for the whole time interval $[S|_G(L(OPT_G)), E|_G(L(OPT_G))]$. But, this also means that he cannot stay the whole time interval at $L(OPT_F)$, and we conclude $OPT_{F \cup \{h\}} < OPT_F$. \square

Now, we have to look at the dimension of the problem. Figure 3.4 shows that the dimension is at least 4.

The proof that the dimension of the problem is at most 4 uses Helly's theorem. A similar argument using Helly's theorem for a related question concerning the weighted Euclidean 1-center problem is described by Megiddo [54].

Theorem 3.7 (Helly's Theorem [64]). *Let C_1, \dots, C_n be convex sets in \mathbb{R}^d , $n \geq d+1$. Suppose that the intersection of every $d+1$ of these sets is nonempty. Then the intersection of all the C_i is nonempty.*

Theorem 3.8. *The dimension of the meeting scheduling problem is at most 4.*

Proof. Let us assume that $B = \{M_{i_1}, \dots, M_{i_b}\}$ is a basis and $b = |B| > 4$ holds. For every 4-element subset W of $\{PRE_{i_1} \cap SUB_{i_1}, \dots, PRE_{i_b} \cap SUB_{i_b}\}$, let T_W denote the maximum length of a vertical segment in the intersection of the sets in W . Note that, by the assumption that there is a solution with $f(x) > 0$, we know that the intersection of the sets in W is nonempty for every W . We define T' to be the minimum T_W obtained for all possibilities of W . Clearly, T' is an upper bound on the length T of a maximum vertical line segment contained in the set of feasible points for the participants in B . In fact, we can even show that $T = T'$ holds. To this end, let SUB'_{i_k} denote the vertical cone with apex angle $\frac{\pi}{2}$ whose uppermost point is its apex $((l_{i_k}^{sub})_X, (l_{i_k}^{sub})_Y, t_{i_k}^{sub} - T')$ for $1 \leq k \leq b$. We observe that all sets in $\{PRE_{i_1} \cap SUB'_{i_1}, \dots, PRE_{i_b} \cap SUB'_{i_b}\}$ are convex and, by definition of T' , the

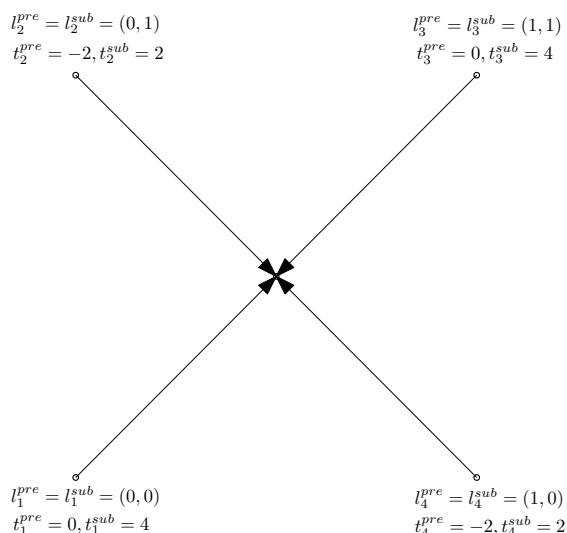


Figure 3.4: Set of 4 participants $\{M_1, M_2, M_3, M_4\}$ and their way to the optimum meeting location. The start of their optimum meeting is determined by participants M_1 and M_3 , while the end is determined by M_2 and M_4 . Removing any participant enables a meeting with longer duration. If one, for example, removes participant M_1 , the optimum meeting location becomes $(1, 1)$ and the duration of the optimum meeting increases by $\sqrt{2} - 1$.

intersection of every 4 of them is nonempty. By Helly's theorem, this implies that the intersection of all sets in $\{PRE_{i_1} \cap SUB'_{i_1}, \dots, PRE_{i_b} \cap SUB'_{i_b}\}$ is nonempty. But a point in this intersection corresponds to a vertical line segment of length at least T' in the intersection of the original cones $\{PRE_{i_1}, \dots, PRE_{i_b}, SUB_{i_1}, \dots, SUB_{i_b}\}$. This means, that $T \geq T'$ and together with $T \leq T'$ we conclude that $T = T'$ holds.

Let V denote a 4-element subset of B such that $f|_V$ and $f|_B$ obtain the same value as maximum. If $f|_V$ obtains its maximum only in one single point x^* , this point must be the optimum meeting point for V as well as for B . But in this case $S|_V(x^*) = S|_B(x^*)$ holds, which means $OPT_V = OPT_B$ and we get contradiction to B being a basis.

Hence, we must consider the case that $f|_V$ obtains its maximum in more than one point. By Lemma 3.4, there is a subset $Z \subset V$ of at most 2 participants such that $f|_Z, f|_V$ and $f|_B$ obtain the same value as maximum. Let x_Z denote the optimum meeting location for the participants in Z and x_B the optimum meeting location for the participants in B . If $x_Z = x_B$, we are done.

By Lemma 3.5, already the participants in Z can schedule meetings of maximum duration only in locations on the line through x_B and x_Z . This means, we can restrict our examination to the plane containing the vertical segments at x_B and x_Z . Figure 3.5 shows the curves resulting from intersecting this plane with the downward

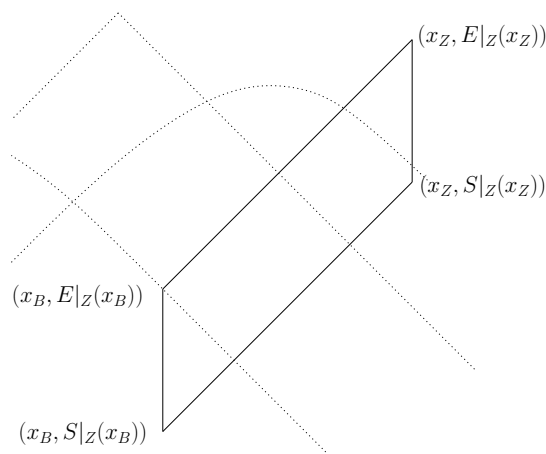


Figure 3.5: The longest meeting duration that is possible for the participants in Z is the same as for the participants in B . The dotted curves correspond to downward cones in $B \setminus Z$.

cones corresponding to participants in $B \setminus Z$. The possible meeting duration at location x_B is the same for the participants in B and for the participants in Z . This implies that all participants in B can reach x_B at $S|_Z(x_B)$, and hence they can also reach x_Z at $S|_Z(x_B) + \|x_Z - x_B\|$, which equals $S|_Z(x_Z)$ by Lemma 3.2. Consequently, the upward cones corresponding to participants of $B \setminus Z$ are not important.

The optimality of x_B ensures that there is a downward cone PRE_{i_j} which contains the point $(x_B, E|_Z(x_B))$ on its boundary and prevents improving the solution by moving from x_B into the direction of x_Z . We conclude, that the optimum meeting for the at most 3 participants in $Z \cup \{M_{i_j}\}$ is not better than for all participants in B , which is a contradiction to B being a basis. \square

Let us recall the primitive operations used for the algorithm by Sharir and Welzl. The optimum solution for at most 5 participants can be found in time $O(1)$. To this end, we could for example use the $O(n \log n)$ algorithm presented in [43, 16, 17]. We conclude that the primitive operations *basis change* and *violation test* can be done in time $O(1)$. Finally, any participant can be used as initial basis.

Now we can state our main result:

Theorem 3.9. *We can compute a meeting of longest duration (or determine that no meeting is possible) for n participants in expected time $O(n)$.*

3.4.4 Allowing participants to travel at different speeds

Now, let us assume that the i th participant M_i travels at speed α_i , where $0 < \alpha_i < \infty$. Then the definition of the function S , which expresses the time of the arrival of the

last participants at location x has to be changed to

$$S(x) := \max_{1 \leq i \leq n} \left(t_i^{pre} + \frac{1}{\alpha_i} \|t_i^{pre} - x\| \right)$$

The function E has to be changed accordingly.

For the geometric interpretation of the problem, the set of feasible points can still be described as the intersection of upward and downwards cones. The only difference to the unit speed case is that the apex angle of the cones belonging to participant M_i traveling at speed α_i becomes $2 \arctan(\alpha_i)$. Due to the different speeds, the projection of the cones in PRE on the XY -plane is a multiplicatively weighted furthest site Voronoi diagram of circles. We do not know whether the $O(n \log n)$ algorithm presented in [43, 16, 17] can be generalized to this case. But it is important for the LP-approach that it is still possible to find the optimum meeting location for at most 5 participants in time $O(1)$ by inspecting every cell of the overlay of the Voronoi diagrams belonging to the projections of the cones in PRE and SUB on the XY -plane.

Another problem for generalizing our results to participants traveling at different speeds is that Lemma 3.2 is no longer true. The following Lemma serves as a substitute.

Lemma 3.10. *If $x_1 \neq x_2$ and both locations x_1, x_2 maximize f , then $S(x_1) \neq S(x_2)$, even if participants travel at different speeds.*

Proof. As shown in the proof of Lemma 3.2, the line segment from $(x_1, S(x_1))$ to $(x_2, S(x_2))$ is completely contained in the boundary of a cone, which implies that it has slope > 0 .

□

For the proof of Lemma 3.3, the only necessary change is to replace the use of Lemma 3.2 by the use of Lemma 3.10. The proof of Lemma 3.4 as well as of Lemma 3.5 directly translate to our generalized problem setting.

There is also a minor change with respect to the proof of Theorem 3.8 and Figure 3.5. Now, with the cones having different angles, even upward cones can prevent us from improving the solution, see Figure 3.6.

Finally, we conclude with the generalized result.

Theorem 3.11. *We can compute a meeting of longest duration (or determine that no meeting is possible) for n participants traveling at different speeds in expected time $O(n)$.*

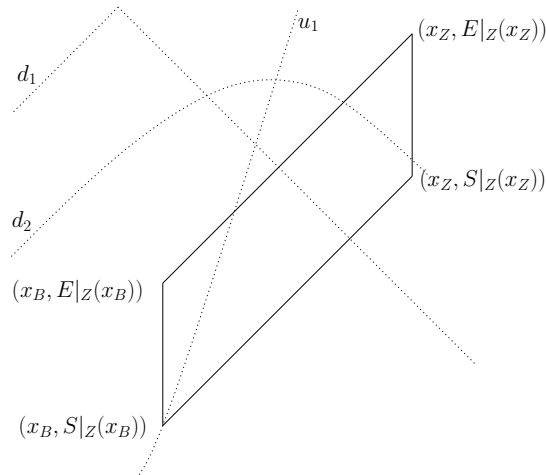


Figure 3.6: The longest meeting duration that is possible for the participants in Z is the same as for the participants in B . The dotted curves d_1, d_2 correspond to downward cones in $B \setminus Z$, while the dotted curve u_1 corresponds to an upward cone.

3.5 Allowing k participants to be absent

Another interesting variant of the meeting scheduling problem is the following. Given a problem instance with n participants and a parameter k , find a meeting point which maximizes the duration for a meeting of at least $n - k$ participants. For example, we could think of a situation where half the participants have a quorum. An optimum meeting for all except k participants can be found by inspecting the optimum meeting point for every subset of 4 participants, but we would like to have a faster algorithm. However, a subquadratic solution in the full range of k seems unlikely, because we will establish that this problem is *3SUM-hard*.

The class of 3SUM-hard problems has been introduced by Gajentaan and Overmars [32], no subquadratic solution for any 3SUM-hard problem is known.

As we already noticed, the meeting scheduling problem for all participants contains the smallest enclosing disk problem. Accordingly, the meeting scheduling problem for the best meeting for $n - k$ of n participants contains the problem to find the smallest disk enclosing at least $n - k$ of n given points in the plane. In [53], it is mentioned that the 3SUM-hardness of the latter problem has been observed by David Eppstein. I would like to thank him for providing me the proof of the following Theorem [28].

Theorem 3.12. *The problem to find the smallest disk enclosing at least q points of a set of n points in the plane is 3SUM-hard.*

Proof. One of the base 3SUM-hard problems introduced by Gajentaan and Overmars is

GEOMBASE: Given a set S of n points in the plane with integer coordinates on the three vertical lines $x = 0$, $x = 1$, and $x = 2$, determine whether there exists a non-vertical line containing three of the points.

We will establish that a subquadratic algorithm for the problem to find a smallest disk enclosing at least q of n given points would also imply a subquadratic algorithm for GEOMBASE. To this end, we consider the dual of GEOMBASE. The dualization maps a point (a, b) to the line $\{Y = aX + b\}$, while this line is mapped to the point $(-a, b)$.

The dual of GEOMBASE is the following. We are given a set L of n lines. The lines have slope 0, 1 or 2 and intersect the y -axis at points with integer coordinates. We have to determine whether there exists a point belonging to 3 of the lines. Let us consider the intersection point p of two lines $l_1 : x \mapsto \alpha_1 x + i_1$ and $l_2 : x \mapsto \alpha_2 x + i_2$, where $\alpha_1, \alpha_2 \in \{0, 1, 2\}$, $\alpha_1 \neq \alpha_2$ and $i_1, i_2 \in \mathbb{Z}$. We assume that no other line of L contains p . The x -coordinate of the intersection point $p = (p_x, p_y)$ equals $p_x = \frac{i_2 - i_1}{\alpha_1 - \alpha_2}$. In all possible cases, this is half of an integer. Clearly, all lines of $L \setminus \{l_1, l_2\}$ having slope α_1 or α_2 have vertical distance at least 1 from p . On the other hand, the y -coordinate of lines from $L \setminus \{l_1, l_2\}$ with the remaining slope $\{0, 1, 2\} \setminus \{\alpha_1, \alpha_2\}$ is half of an integer at p_x . Since these y -coordinates are different from p_y by assumption, we conclude that every line different from l_1 and l_2 has, at x -coordinate p_x , vertical distance at least $\frac{1}{2}$ from p . The fact that we are only dealing with lines of slope 0, 1 and 2 makes it easy to verify that the vertical distance from an arbitrary point $r \in \mathbb{R}^2$ to such a line l is at most $\sqrt{5}$ times the Euclidean distance from r to l .

To summarize it: every intersection point of two lines of L has distance at least $\frac{1}{2} \frac{1}{\sqrt{5}} > \frac{1}{5}$ from its third nearest neighboring line, if there is no point contained in 3 lines.

Let B denote a bounding box of the intersection points of the lines of L . We define y_{\min} and y_{\max} to be the minimum and maximum y -coordinate where lines from L pass the y -axis. Any two intersecting lines $l_1 : x \mapsto \alpha_1 x + i_1$ and $l_2 : x \mapsto \alpha_2 x + i_2$ have the x -coordinate of their intersection point at $\frac{i_2 - i_1}{\alpha_1 - \alpha_2}$. Since $\alpha_1 - \alpha_2 \in \{-1, 1\}$, this x -coordinate is somewhere between $-(y_{\max} - y_{\min})$ and $y_{\max} - y_{\min}$. Of course, this directly implies bounds for the y -coordinate of the intersection point. Hence, we can construct the bounding box B of the intersection points of all lines of L in time $O(n)$.

Now, we are ready for the main step of the proof. We approximate the relevant part of every line by the intersection of two large disks. To this end, we choose a radius R large enough that we can find two disks of radius R for every line $l \in L$, such that

- each of the two half-planes induced by l contains one of the disk centers,
- both disks contain $l \cap B$,
- both disks together cover B , and
- the boundaries of the disks stay within distance less than $\frac{1}{10}$ from l within B .

We denote the set of the $2n$ disks as C and the set of the $2n$ disk centers as D .

There exists a point belonging to 3 of the lines of L , if and only if there is some point p contained in $n + 3$ of the disks of C . However, there is a point p contained in $n + 3$ of the disks of C , if and only if there exists a disk of radius $\leq R$ containing at least $3n + 3$ of the points of D .

The latter problem can be decided by an algorithm which is able to compute the minimum radius of a disk containing at least $n + 3$ of the points of D . The resulting radius is $\leq R$, if and only if the answer to the corresponding GEOMBASE instance is 'yes'. The transformation from the GEOMBASE instance to the smallest disk enclosing at least q of n points instance can be done in linear time. \square

3.6 Maximizing the number of participants

If it is not possible for all participants to schedule a common meeting, it might be the best to seek a meeting point where as many participants as possible can schedule a common meeting. We are only interested in maximizing the number of participants, the meeting length is not important for this problem variant.

Once again, 3SUM-hardness is involved. Let us consider an instance where all $l_i^{pre} = l_i^{sub}$, all $t_i^{pre} = 0$ and all $t_i^{sub} = T$ for some constant T . Clearly, any set of participants who are able to have a common meeting at all, can also have this common meeting at time $\frac{T}{2}$. If we intersect the cones corresponding to the participants with the plane corresponding to that time point $\frac{T}{2}$, the problem becomes to find, for a given set of disks of equal radii in the plane, a point which is contained in a maximum number of disks. An algorithm which is able to compute such a point could also be used to decide whether there exists point p contained in $n + 3$ of the disks of the set C defined in the proof of Theorem 3.12. We conclude that the problem to determine a point contained in a maximum number of disks is 3SUM-hard, which has also been shown by Aronov and Har-Peled [7].

Unfortunately, we are not aware of an algorithm for maximizing the number of participants in quadratic time. However, it is easy to solve this problem in time $O(n^3 \log n)$.

Theorem 3.13. *For n participants, a meeting point maximizing the number of participants who can schedule a common meeting can be found in time $O(n^3 \log n)$.*

Proof. The $2n$ cones corresponding to the n participants divide the 3-dimensional space into different regions. Each region allows a meeting for the same subset of participants. For each region, let us consider a lowest point of the region. This lowest point is either an apex of an upward cone or it belongs to the intersection of two upward cones.

We can inspect all apexes of upward cones and compute the corresponding maximum number of participants in time $O(n^2)$. Afterwards, we inspect the $O(n^2)$ intersection

curves for each pair of upward cones. For each such curve γ we do the following:

Since γ is a curve of constant degree, we know that γ has at most $O(n)$ intersections with the $2n$ cones belonging to the participants. Let us compute all these intersection points and sort them according to their order on γ . Afterwards, we can sweep over the curve γ , and for each intersection with a cone, we know whether this intersection implies that we are gaining or losing one participant. After the sweep, we know the maximum number of participants which is possible on γ .

For each curve γ , we determine the maximum number of participants in time $O(n \log n)$. Since we have to consider $O(n^2)$ curves, our approach leads to an $O(n^3 \log n)$ algorithm. \square

3.7 Multiple meeting locations

Assume, we are allowed to have meetings at k different locations instead of just one location. The goal is to maximize the common meeting time for all participants, where every participant can travel to one out of the k locations. We still want to find a common meeting within the same time range for all participants. A motivation could be that all locations are equipped for a common video conference.

Let us first consider the situation, where all $l_i^{pre} = l_i^{sub}$, all $t_i^{pre} = 0$ and all $t_i^{sub} = T$ for some sufficiently large constant T . In this special case, the best for each participant is to travel to his nearest meeting location. The problem becomes the well known:

CENTER k -CLUSTERING-PROBLEM: Given a set S of n points in d -dimensional Euclidean space and an integer k , find a partition of S into k clusters and a center for each cluster, such that the maximum distance from any point of S to its cluster center is minimized.

For dimension $d \geq 2$, Fowler *et al.* [30] proved center k -clustering to be NP-complete. The meeting scheduling problem at k different locations belongs to the class NP, because the partition of the participants into the k different groups can be used as certificate.

Finally, we conclude

Theorem 3.14. *To find the optimum meeting for n participants at k different meeting locations is NP-complete.*

3.8 More than 2 previously scheduled meetings

In this section, we want to get rid of the assumption that we have to take into account only 2 meetings already scheduled for each participant. Instead, we allow

every participant to schedule the new meeting somewhere between up to at most k meetings already scheduled.

Let us assume that participant M_i has his first meeting at location l_i^1 , where he has to stay until time $t_{i,1}^{pre}$. His second meeting is at location l_i^2 , starts at time $t_{i,2}^{sub}$ and lasts until time $t_{i,2}^{pre}$, and so on. His last meeting starts at location l_i^j at time $t_{i,j}^{sub}$, where $j \leq k$.

How to schedule a meeting of longest possible duration in between the other meetings of all participants? Can we once again use the concept of LP-type problems?

In fact, the generalized problem is still an LP-type problem. Clearly, any meeting which is possible for a set of participants is also possible for every subset. What about Axiom 2? A necessary condition for participant M_i to be able to come to a new meeting from time t_1 to time t_2 is that there exists some index $h \in \{1, \dots, k-1\}$ such that t_1 and t_2 are both contained in the interval $[t_{i,h}^{pre}, t_{i,h+1}^{sub}]$. But within this interval, we can express the feasibility of the meeting with respect to M_i in the condition that the corresponding vertical segment is contained in the intersection of just one upward and one downward cone.

The optimum meeting for all participants corresponds to the optimum solution for an instance of the problem where every participant has only 2 meetings. Two optimum meetings correspond to two vertical segments, which have the same length and the same time range. For this time range, the problem can be stated within the original problem setting where every participant has only 2 meetings scheduled. This means that the proof of Axiom 2 from Theorem 3.6 inherits to the generalized problem setting.

Now, that we know that our generalized problem is still LP-type, does this also mean that we can once again solve in expected linear time using the algorithm of Welzl and Sharir?

This time, the answer to this question is 'no', because the dimension is no longer restricted by 4 or by any other constant. In the following, we describe an instance of n participants, where removing any of them improves the solution. All meeting location are at the same point, hence we do not have to take into account travel times. Every participant has n time intervals, where the new meeting could be scheduled. All these time intervals have length 1 or 2. Figure 3.7 illustrates the time intervals: the i -th row shows the i -th participant's time intervals. Clearly, the best meeting for all participants has duration 1, whereas the best meeting for any proper subset with at least one participant has duration 2.

As we already observed, for each fixed time point t , finding a meeting of maximum duration whose time interval contains t can be done as in the situation where every participant has only 2 meetings scheduled. This fact is used for the following algorithm. We define $T(M_i) := [t_{i,1}^{pre}, t_{i,2}^{sub}] \cup [t_{i,2}^{pre}, t_{i,3}^{sub}] \cup \dots$ to be the set of free time intervals for participant M_i .



Figure 3.7: Participants having these free time intervals are a basis of cardinality n .

Algorithm:

- Compute $C := \bigcap_{i=1}^n T(M_i)$.
- For each connected component of C determine the optimum solution using the Welzl/Shair-algorithm.
- Return the best value found.

The above algorithm returns the optimum solution for the meeting scheduling problem with multiple time intervals. What about the performance? Computing $\bigcap T(M_i)$ can be done in $O(kn \log(kn))$ by sweeping over the sorted list of all interval endpoints. For the remaining part of the algorithm, the expected running time is $O(vn)$, where v denotes the number of connected components of C .

We assume that every participant has at most k meetings, which lead to at most $k - 1$ free time intervals for each. All participants' free time intervals have at most $2n(k - 1)$ endpoints, which are the only possible endpoints for the connected components of C . If we sort all endpoints of all intervals, the first $n - 1$ as well as the last $n - 1$ are not important for the boundary of C . Since we need two of the remaining interval endpoints for every connected component of C , we conclude that C contains at most $1 + n(k - 2) \in O(kn)$ connected components.

Figure 3.8 shows an instance, where this number of connected component really occurs.

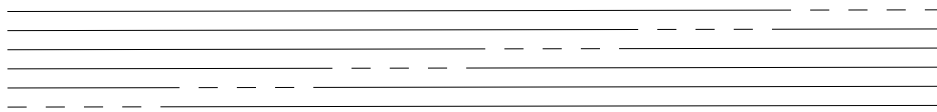


Figure 3.8: n participants, each of them has $k - 1$ free time intervals. The number of connected components of $\bigcap T(M_i)$ equals $1 + n(k - 2)$.

Now, that we know the exact upper bound on the number of connected components, we can state the main result of this section:

Theorem 3.15. *We can compute a meeting of longest duration (or determine that no meeting is possible) for n participants in expected time $O(kn^2 + kn \log(kn))$, if each participant has at most k meetings already scheduled.*

3.9 Conclusion

We conclude that the meeting scheduling problem, as stated here, can be solved in $O(n \log n)$ time by an algorithm that is based on geometrical observations and by an expected linear-time randomized algorithm within the LP-type framework. Many interesting questions are still under investigation. For example we want to generalize our results to other metrics. Another question is whether a deterministic linear-time algorithm for LP-type problems can be applied to our problem. In principle such an algorithm has been described by Chazelle and Matoušek [23], but it needs more restrictive assumptions than the algorithm by Sharir and Welzl we have used in this thesis.

For the 3SUM-hard problem to maximize the number of participants, we presented an $O(n^3 \log n)$ algorithm. Is it possible to solve this problem faster? The same question arises for the problem variant where each participant has up to k previously scheduled meetings.

Chapter 4

Traveller's Problem

4.1 Introduction

Motion planning in dynamic environments is one of the challenging problems in computational geometry. A classical, and important question is how to avoid collision with obstacles that move in time. Motion planning for a point-shaped robot in an environment where each obstacle is a convex polygon moving in a fixed direction at constant speed is analyzed in [31]. A method to compute the trajectories of a robot moving in a time-varying environment, using velocity information to directly determine potential collisions, is presented in [29]. If the movement of the obstacles is unpredictable, the area which has to be avoided can be modeled by discs growing over time [52]. In [6], the authors analyze the problem of finding a maximum number of disjoint paths for unit disks moving amidst static or dynamic obstacles and present a pseudopolynomial-time dual-approximation algorithm for their problem.

We take a different point of view. Instead of the situation that the moving objects are obstacles, we introduce a new motion planning problem where a *traveller* can use the moving objects as a means of transportation.

A *carrier* in dimension d is a non-empty intersection of finitely many closed half-spaces. Thus, in dimension 2, a carrier can be a point, a line segment, a half-line, a line, a possibly unbounded convex polygon, or the plane itself. Each carrier is assigned its own velocity vector, causing it to move linearly at constant speed. The traveller is modeled as a point, p . At time $t = 0$, his journey begins at a start point, s , which is located on some carrier C_s . Like a passenger on board a vessel, traveller p will be moved along with C_s . In addition, he can walk on C_s (and any other carrier) at maximum speed v , called *innate speed* for short, as long as he does not fall off. If, at some time $t > 0$, traveller p is located in the intersection of C_s and some other carrier, C , he may decide to change from C_s to C , and continue his journey by C . The traveller's ultimate goal is to reach a goal point, g , located on some carrier C_g .

We observe that our definition of changing the carrier allows for the following imple-

mentations. First, the traveller can change at some point where two carriers touch each other, since such touch point belongs to either carrier. Second, we could introduce a special carrier C_0 that equals the whole plane and does not move. Then the traveller could, in our model, get off his current carrier anytime and wait for another carrier to arrive or, if $v > 0$, walk some distance in the plane, and board another carrier. In this model, start point s and goal point g could also be located in the plane, instead of being fixed to moving carriers.

In general it is not clear if g can be reached from s at all. Thus, we are interested in the following questions.

- Given initial positions for the carriers at time $t = 0$, is it possible for the traveller to reach g when starting from s ?
- If so, what is the quickest way to get there, using only the carriers for transportation?

This model could be generalized in many ways. Instead of the quickest way, one could ask for the journey that minimizes the traveller's walking. Using a carrier could incur a cost; then a cheapest s -to- g path would be of interest. Moreover, there is some affinity to the highway systems on which city Voronoi diagrams [1, 35, 9] are based. We could model a straight, unbounded highway lane by a line that moves in axial direction. The same holds for navigation in currents near the shore [36]. However, our model differs from these settings in that carriers can move in arbitrary directions, independent of their shapes. We restrict ourselves to the simple model introduced above, and study the question of finding a quickest connection.

We will establish that the complexity of our problem arises from two different points of view.

The first one is an *analytical complexity*, caused by the following surprising fact. There are very simple scenes in two-dimensional space where the goal can be reached in finite time, but only by an infinite number of carrier changes. The situation is even worse in three-dimensional space where the set of carrier change time points may require infinitely many accumulation points for the traveller to reach his goal. As a consequence of the analytical complexity, our arguments will involve concepts from calculus as convergence of sequences or accumulation points.

The other big challenge besides the analytical complexity is the *combinatorial complexity* of the problem. Even if the number of carrier changes is known to be linear, the two-dimensional problem is NP-hard. However, NP-hardness is not at all the end of the story. In high dimensions, the surprising result is that the problem is undecidable, even if the number of carrier changes is known to be finite.

Our problem fits well in the context of *hybrid automata* [58, 4]. These are systems combining discrete and continuous components. Each hybrid automaton can be expressed as a directed graph of discrete states and transitions, augmented with

several real-valued continuous variables. While in a discrete state, the evolution of the continuous variables is governed by a differential equation assigned to this state. Furthermore the continuous variables determine whether it is allowed to change from one discrete state into another one. The hybrid automaton corresponding to our model has the *Zeno* property (see *e.g.* [41]), which means that an optimum traveller path may need an infinite number of carrier changes in a finite time interval. The reason why *Zeno* systems are named as they are is that the situation is similar to the paradox of Achilles and the tortoise. The reachability question for hybrid automata is whether a certain goal configuration is reachable from a certain initial configuration. A lot of work has been done during the last years in finding the boundary between decidable and undecidable hybrid systems [8]. For many systems the decidability or the undecidability has been proven. On the other hand, there are also a lot of systems where this question is still an open problem.

Now, as a warm-up example for Traveller’s Problem, let us consider a special case that resembles the well-known “Frogger” game.

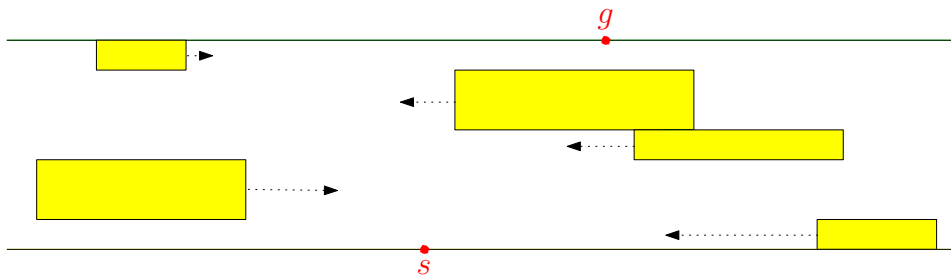


Figure 4.1: Barges on a river.

Our traveller wants to cross a river whose banks are modeled by two static lines containing s and g . The only other carriers are n rectangular barges that move up- or downstream at m different speeds, such that tracks of neighboring barges touch; see Figure 4.1. As with Frogger, the traveller can move on each carrier at maximum L_1 speed v .

Theorem 4.1. *Suppose that the n barges are running at m different speeds. Then Traveller’s Problem can, in this case, be solved in time $O(n \log m)$.*

The proof of Theorem 4.1 uses the continuous Dijkstra paradigm [56] and an idea of [48]. It can be found in [48] and [15].

Let us return to the general Traveller’s Problem and give its formal problem description in the following section.

4.1.1 Problem description

An instance of *Traveller’s Problem* consists of:

- a dimension d and a metric $dist$ on \mathbb{R}^d
- a finite set C of carriers, $|C| = n$
- for each $c \in C$, a (possibly unbounded) convex polyhedron $R_0(c)$ corresponding to c at time $t = 0$
- for each $c \in C$, a velocity vector \vec{c}
- the traveller's start location s at time $t = 0$
- the goal location g at time $t = 0$ and the velocity vector \vec{g} of the goal
- the traveller's maximum innate speed v .

A *traveller path* (I, f, A, g) is given by:

- a closed and bounded interval I
- a continuous map $f : I \rightarrow \mathbb{R}^d$
- a set $A \subset I$ which is countable and closed, accordingly all connected components of $I \setminus A$ are open intervals
- a map $g : I \setminus A \rightarrow C$ which is constant on each connected component of $I \setminus A$, $g((a, a')) \neq g((a', a''))$, if (a, a') and (a', a'') are connected components of $I \setminus A$

such that:

- If (a, a') is a connected component of $I \setminus A$ and $g(t) = c \forall t \in (a, a')$, then $\forall t \in (a, a')$:
 - $dist(f(t), f(a) + (t - a)\vec{c}) \leq (t - a)v$, and
 - $f(t) \in R_0(c) + t\vec{c}$.

For $I = [0, T]$, the path is said to *connect the start location with the goal location*, if $f(0) = s$ and $f(T) = g + T\vec{g}$.

Given an instance of Traveller's Problem, let T' denote the infimum of all travel times of traveller paths connecting the start location with the goal location.

The task of Traveller's Problem is to decide whether a path from the start location to the goal location exists. If this is the case we have $T' < \infty$ and the task is also to find, for any $\varepsilon > 0$, a traveller path of travel time at most $T' + \varepsilon$.

The reason for allowing infinitely many carrier changes and using the infimum instead of the minimum in the definition above are subjects of the following section.

4.1.2 Surprising properties of traveller paths

Surprisingly, there are very simple scenes where the goal can be reached in finite time, but only by an infinite number of carrier changes. An example is shown in Figure 4.2. The three line segments A, B, C are very long, but bounded. Their velocity vectors have large axial and small lateral components. This causes the segments to intersect in point z at some time t_z . The fast point-shaped carrier D will pass through z at time t_z , too, and then speed on to meet carrier E that consists just of the goal point, g . The traveller sets out from point s on A . In order to reach g , he needs to

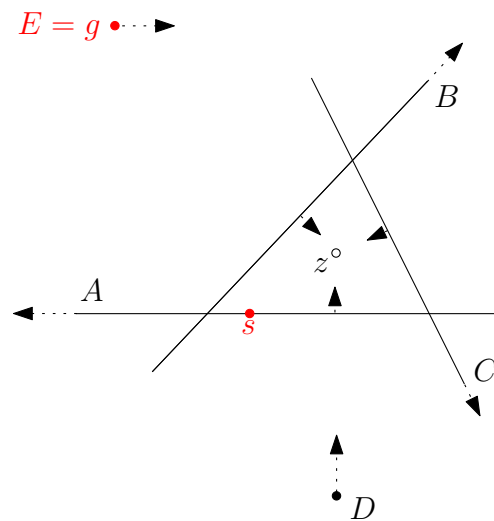


Figure 4.2: The traveller must pass through the cycle A, B, C infinitely often before he can board carrier D .

catch carrier D , because A 's upward movement is too slow. The only occasion when one of A, B, C contains D is at time t_z in point z . Until then, the traveller must keep cycling through the ever contracting triangle formed by A, B, C . The following argument establishes that cycling this triangle an infinite number of times might be necessary.

Suppose that the traveller's innate speed is zero, and that in Figure 4.2, point z can be reached at time t_z when A, B, C intersect in z after only a finite number of carrier changes. Consider the last change, and suppose it was from carrier C to carrier A . Clearly, it takes place before time t_z (or it would be unnecessary, because the traveller has already arrived at z). So imagine the traveller standing at the intersection point—let us call it p —of C and A in Figure 4.2. He can only reach z without further carrier changes if the speed vector of A , that is, the sum of the components parallel and orthogonal to A , points from p to z . But that may never happen, for example, if the triangle is always equilateral, and centered at z , and if all the velocity vectors form angles < 30 degrees with their carriers.

We obtain the following lemma.

Lemma 4.2. *There are two-dimensional instances of Traveller's Problem where the traveller can reach the goal location, but an infinite number of carrier changes is necessary to get there.*

If the bounded set A has infinitely many points, then it has an accumulation point by Bolzano-Weierstrass. In the above example, this accumulation point is the time point t_z . Now, we sketch a three-dimensional situation which is even worse than just one accumulation point of A . We introduce an instance where A requires infinitely many accumulation points for the traveller to reach the goal.

Lemma 4.3. *There are instances of Traveller's Problem where the goal can be reached, but the set A of time points of carrier changes requires infinitely many accumulation points to do so.*

Proof. We consider the 3-dimensional instance shown in Figure 4.3.

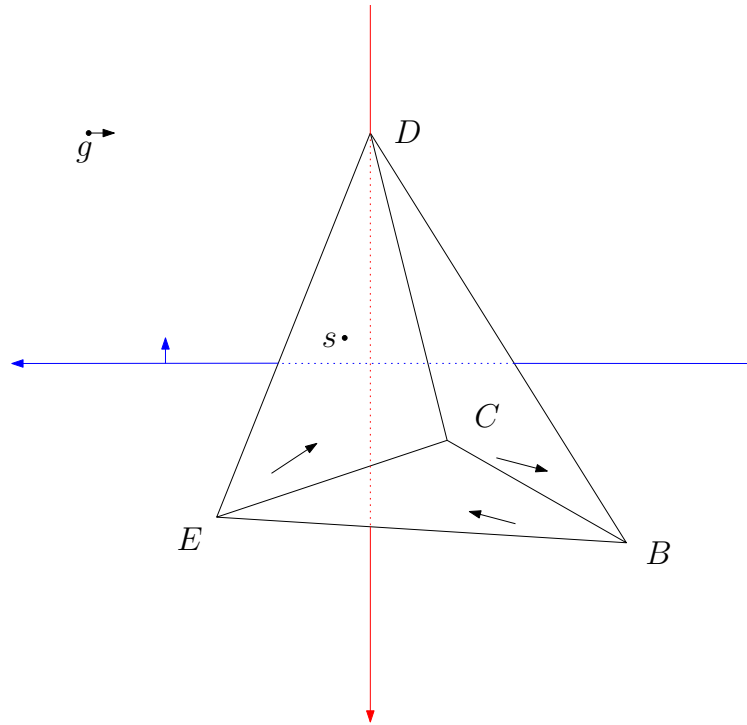


Figure 4.3: Infinitely many accumulation points.

The points B, C, D, E are the vertices of a tetrahedron. Point C is closer to the viewer than the plane through B, D, E . Every plane through CDE , BCD and BDE is a carrier. Only the tetrahedron triangles of these carriers is shown in the picture.

The black arrows indicate the velocity vectors of these carriers. Each plane moves into a direction such that its position in 3-dimensional space keeps the same. We can think of these carriers as unbounded conveyor belts.

The traveller located at start point s on the carrier through CDE can use this carrier until he reaches the common edge with the carrier through BCD . Then he can change to this carrier until he reaches the common edge with BDE and so on. Each round where the traveller uses all three tetrahedron carriers reduces his distance to the point D by a constant factor. After taking the round infinitely often, the traveller reaches the point D . From there on he can use the red carrier until he changes to the blue one which brings him back to a point on the boundary of the tetrahedron. Then he can again use the tetrahedron carriers infinitely often to reach point D and so on.

Since the blue carrier moves upwards slowly, the time difference between consecutive time points when the traveller reaches D gets smaller and smaller. Each time point the traveller reaches D corresponds to an accumulation point of A .

After reaching D infinitely often, the traveller can be at D simultaneously with the blue carrier. If the goal carrier intersects the other ones only at that moment, the traveller has to perform his path this way.

This means that the set A of time points of carrier changes contains infinitely many accumulation points. These accumulation points themselves accumulate at that time point when the traveller reaches the goal. \square

We continue to illustrate the complexity of traveller paths in three-dimensional space by another example. This time, the purpose is to show that the infimum travel time is not a minimum in general.

Lemma 4.4. *There are instances of Traveller's Problem where the infimum travel time to the goal location is $T' < \infty$, but not traveller path reaching the goal at time T' exists.*

Proof. We consider a three-dimensional instance containing 5 carriers. Three of them, so-called *attic carriers*, are shown in Figure 4.4.

Each pair of roof ridges of attic carriers intersects at an angle of $\frac{2\pi}{3}$. The velocity vectors of the attic carriers are chosen such that the intersection point z of their roof ridges stays the same all the time. The scene contains two further carriers. One is the *elevator carrier* which fills the whole 3-dimensional space and moves downwards at speed 1. The last carrier is the point-shaped goal carrier which starts at a point over z and moves downwards vertically at speed 1. Let T' denote that time point when the goal carrier reaches z . We consider traveller paths which start at z and have maximum innate speed 0.

We show the following three statements:

- (1) All traveller paths to the goal require travel time $\geq T'$. — None of the carriers

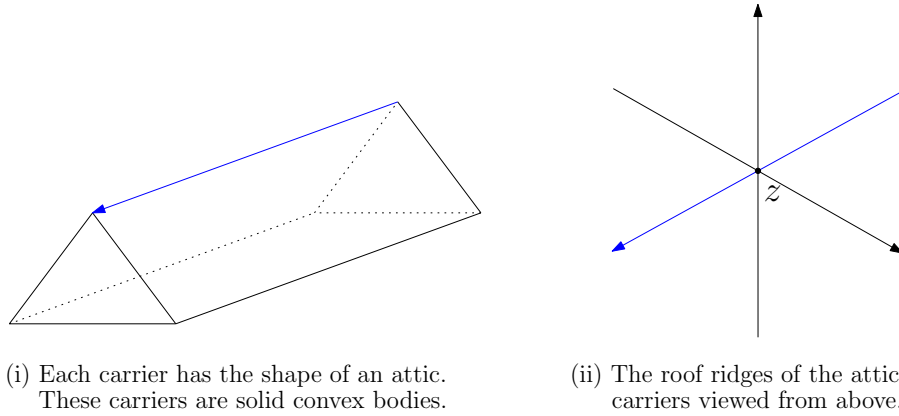


Figure 4.4: Attic carriers.

enables the traveller to gain height. Hence, the traveller cannot reach the goal location before the goal carrier has reached the height of z which happens at time T' .

- (2) There are traveller paths to the goal whose travel times come arbitrarily close to T' . — Let the traveller start his trip by using one of the attic carriers for duration ε . Afterwards, he can use the elevator carrier for duration $c\varepsilon$ (with c as a suitable constant) until he reaches the boundary of both other attic carriers. From then on, he can stay in the intersection of all three attic carriers until the end of his trip. If he uses both attic carriers different from his start carrier for duration ε , he reaches a point directly under z . Using the three attic carriers in cyclic order and staying in their common intersection enables him to meet the goal location at that moment when the goal has reached his height. The total travel time of the trip is $T' + O(\varepsilon)$.
- (3) No traveller path reaches the goal location at time T' . — By Lemma 4.5, which will be discussed in Section 4.2, we can choose a subinterval of the time where only a finite number of carrier changes happens. If the elevator carrier is used during this subinterval, then the height of the traveller will be below the roof ridges afterwards. Otherwise, we have at least one carrier change from one attic carrier to another attic carrier which happens at place not on the vertical line through z . This carrier change requires both carriers to overlap which means that also in this case the height of the traveller has to be below the roof ridges. Afterwards the traveller cannot increase his height which means that he cannot reach the goal at time T' .

Thanks to statements (1)-(3), the infimum travel time T' is not a minimum travel time. □

4.1.3 Overview of the results of this chapter

Traveller paths containing an infinite number of carrier changes are difficult to handle. In certain circumstances, Lemma 4.5 which is introduced in Section 4.2 enables us to restrict ourselves to subintervals containing only finitely many carrier changes.

The ease of the solution for our warm-up example, the “Frogger” game, was owed to the special situation. In Section 4.3 we show that the general Traveller’s Problem is undecidable in dimension 8 or higher, employing a recent result on affine mappings [10].

Of all smaller dimensions, the case of the Euclidean plane is of particular interest. In Section 4.4 we provide the following results. First, we show the problem to be NP-hard, even if all carriers are vertical line segments. Not even a constant factor approximation is possible, unless $P=NP$. Bounding the complexity of the problem from above is made difficult by the possibility of infinitely many carrier changes. Despite this complication, we can show that, in dimension two, Traveller’s Problem is decidable if the carriers are lines or line segments. By discretising time and space, we also provide a pseudo-polynomial approximation algorithm for the case of bounded, but otherwise arbitrary, convex carriers in the plane. To this end, we relax our original model by increasing the traveller’s innate speed by ε and the extensions of the carriers in all directions by μ . If an s -to- g path of duration $\leq W$ exists in the original model, then, in the relaxed model, a slightly longer journey can be computed in time

$$O\left(n^2 \log(L) W \frac{G}{H}\right)$$

where L denotes the total number of edges of the n carriers, G equals the maximum carrier diameter squared times the maximum speed to the 5th power, and $H = \varepsilon^4 \mu^3$.

Finally, in Section 4.5, we discuss the possibility of uncountable many traveller paths and construct a three-dimensional instance allowing the traveller to reach exactly the points of the famous Cantor set.

We presented some of our discoveries about Traveller’s Problem first in [14, 15].

4.2 Subintervals with finitely many carrier changes

Traveller paths containing an infinite number of carrier changes—which means that $|A| = \infty$ —are difficult to handle. However, the following Lemma shows that it is always possible to consider such paths restricted to subintervals containing only finitely many points of A . I would like to thank Thomas Böhme and Friedrich Regen [19] for providing me the proof of the following Lemma.

Lemma 4.5. *Let $A \subset \mathbb{R}$ with $|A| = \infty$ denote a set, which is bounded, closed and countable. Then, for any natural number N , there exists a closed interval $J \subset \mathbb{R}$, such that $N \leq |A \cap J| < \infty$.*

Proof. Since A is bounded and infinite, there exists at least one accumulation point of A . Furthermore, A contains all of its accumulation points, because A is a closed set.

We define

$$\hat{A} := \{a \in A : a \text{ is accumulation point of } A\} \subset A$$

We claim that \hat{A} contains an element \hat{a} , which is not an accumulation point of \hat{A} .

Let us for contradiction assume that all elements of \hat{A} are accumulation points of \hat{A} . Clearly, in this case the number of elements of \hat{A} is infinite. It is well known from topology that the set \hat{A} is a closed set. Let t_0 denote the minimum of \hat{A} and t_1 denote the maximum. The difference $\Delta := t_1 - t_0$ is positive, because there is more than one element in \hat{A} .

Now, we define two sets A_0 and A_1 .

$$A_0 := \{a \in A : |a - t_0| \leq \frac{\Delta}{3}\}$$

$$A_1 := \{a \in A : |a - t_1| \leq \frac{\Delta}{3}\}$$

Notice that the sets A_0 and A_1 have a positive distance of at least $\frac{\Delta}{3}$ from each other. By assumption, the point t_0 is an accumulation point of \hat{A} . This implies that there is an infinite number of elements of \hat{A} in the interval $[t_0 - \frac{\Delta}{3}, t_0 + \frac{\Delta}{3}]$. All these accumulation points belong to A_0 . Hence, the set A_0 is again bounded, infinite, countable and closed. This means we can repeat the above splitting which leads to subsets $A_{0,0}$ and $A_{0,1}$. Accordingly, also the set A_1 can be split into two subsets $A_{1,0}$ and $A_{1,1}$.

By iterating this splitting, we obtain a set A_b for every finite binary sequence b . Each set A_b is bounded, infinite, countable and closed. If b_1 and b_2 denote two different binary sequences of the same length, then the sets A_{b_1} and A_{b_2} have positive distance, because each splitting process creates two subsets of positive distance.

Let b' denote an infinite binary sequence. For each finite subsequence $s_m(b')$ containing only the first m digits of b' , we choose an arbitrary point $h_m \in A_{s_m(b')}$. For $m \rightarrow \infty$, the sequence of points h_m converges, because each splitting process divides any set of diameter D into two subsets whose diameter is at most $\frac{2}{3}D$. Since A is closed, the limit point $\lim_{m \rightarrow \infty} h_m$ belongs to A . This way, we assign every infinite binary sequence to one point of A . Different infinite binary sequences are assigned to different points of A , because each splitting process creates two subsets of positive distance. Consequently, the cardinality of A is at least the cardinality of the set of all infinite binary sequences, which is uncountable. This contradicts the precondition that A is countable.

We conclude that \hat{A} contains an element \hat{a} which is not an accumulation point of \hat{A} .

Consequently, we can choose an $\varepsilon > 0$, such that $[\hat{a} - \varepsilon, \hat{a} + \varepsilon] \cap \hat{A} = \{\hat{a}\}$. The point \hat{a} is the only accumulation point of A in the interval $[\hat{a} - \varepsilon, \hat{a} + \varepsilon]$.

For $0 < \mu < \varepsilon$, both intervals $[\hat{a} - \varepsilon, \hat{a} - \mu]$ and $[\hat{a} + \mu, \hat{a} + \varepsilon]$ contain only finitely many points of A . However, for $\mu \rightarrow 0$, at least one of both cardinalities gets arbitrarily large. \square

4.3 Higher dimensions

Now we show that Traveller's Problem is undecidable in dimension $d \geq 8$. To this end we employ the following recent result by Bell and Potapov [10]. Let us recall that a two-dimensional affine mapping over \mathbb{Q} acts on (x, y) as follows. First, (x, y) is multiplied by a 2×2 matrix with rational coefficients, then a rational vector (v, w) is added to the result.

Theorem 4.6. *The following problem is undecidable [10].*

FIVE-AFFINE-MAPPINGS-REACHABILITY:

Given five affine mappings f_1, f_2, \dots, f_5 from \mathbb{Q}^2 to \mathbb{Q}^2 and two rational vectors $q = (x, y)$ and $q' = (x', y')$. Is there a finite product of mappings from $\{f_1, f_2, \dots, f_5\}$ that maps q to q' ?

Their elegant proof is by reduction from Post's correspondence problem [63], replacing letters with 2×2 matrices that form a free semigroup. Now we use this result to prove that Traveller's Problem with innate speed zero is undecidable in dimension $d \geq 8$.

How can we model the effect of five affine mappings in the plane by moving carriers? Let us start with only one affine mapping h .

$$\begin{aligned} h : \mathbb{Q}^2 &\rightarrow \mathbb{Q}^2 \\ (e, f) &\mapsto (\alpha e + \beta f + \gamma, \lambda e + \mu f + \nu) \end{aligned}$$

We will create a scene of moving carriers in 9-dimensional space which has the same effect as h . The coordinate axes are denoted by the variables q, r, \dots, y . The qr -plane corresponds to the plane where h is defined, which means that our goal is that the carriers can be used for a trip from $(e, f, 0, \dots, 0)$ to $(\alpha e + \beta f + \gamma, \lambda e + \mu f + \nu, 0, \dots, 0)$.

The scene contains 20 carriers. One of them is the base carrier C_0 which does not move and equals the qr -plane. The location and velocity vectors of all carriers are shown in Figure 4.1. All carriers are affine subspaces of \mathbb{R}^9 moving into a direction such that their location keeps the same all the time. This means that we can also think of the carriers as infinite conveyor belts.

We have to specify the meaning of the ± 1 terms occurring in some of the velocity vectors. The velocity vector of C_2 equals $(0, 0, +1, 0, 0, 0, 0, 0, 0)$, while the velocity vector of C_3 is $(0, 0, -1, 0, 0, 0, 0, 0, 0)$. Accordingly, the ± 1 terms have to be interpreted for the carrier pairs $C_{5,6}$, $C_{8,9}$ and $C_{11,12}$. The situation is different for $C_{14,15}$

C_0 contains points of form: velocity vector:	$(q, r, 0, 0, 0, 0, 0, 0, 0)$ $(0, 0, 0, 0, 0, 0, 0, 0, 0)$
C_1 contains points of form: velocity vector:	$(q, r, 0, 0, 0, 0, 0, 0, y)$ $(0, 0, 0, 0, 0, 0, 0, 0, 1)$
C_2, C_3 contain points of form: velocity vectors:	$(q, r, s, 0, 0, 0, 0, 0, 1)$ $(0, 0, \pm 1, 0, 0, 0, 0, 0, 0)$
C_4 contains points of form: velocity vector:	$(q, r, \alpha q + \beta r + \gamma, 0, 0, 0, 0, x, 1 - x)$ $(0, 0, 0, 0, 0, 0, 0, 1, -1)$
C_5, C_6 contain points of form: velocity vectors:	$(q, r, s, t, 0, 0, 0, 1, 0)$ $(0, 0, 0, \pm 1, 0, 0, 0, 0, 0)$
C_7 contains points of form: velocity vector:	$(q, r, s, \lambda q + \mu r + \nu, 0, 0, 0, 1, y)$ $(0, 0, 0, 0, 0, 0, 0, 0, 1)$
C_8, C_9 contain points of form: velocity vectors:	$(q, r, s, t, 0, 0, 0, 1, 1)$ $(\pm 1, 0, 0, 0, 0, 0, 0, 0, 0)$
C_{10} contains points of form: velocity vector:	$(0, r, s, t, 0, 0, w, 1 - w, 1 - w)$ $(0, 0, 0, 0, 0, 0, 1, -1, -1)$
C_{11}, C_{12} contain points of form: velocity vectors:	$(0, r, s, t, 0, 0, 1, 0, 0)$ $(0, \pm 1, 0, 0, 0, 0, 0, 0, 0)$
C_{13} contains points of form: velocity vector:	$(0, 0, s, t, 0, 0, 1, 0, y)$ $(0, 0, 0, 0, 0, 0, 0, 0, 1)$
C_{14}, C_{15} contain points of form: velocity vectors:	$(q, 0, s, t, 0, 0, 1, 0, 1)$ $(\pm 1, 0, \pm 1, 0, 0, 0, 0, 0, 0)$
C_{16} contains points of form: velocity vector:	$(q, 0, 0, t, 0, 0, 1, x, 1 - x)$ $(0, 0, 0, 0, 0, 0, 0, 1, -1)$
C_{17}, C_{18} contain points of form: velocity vectors:	$(q, r, 0, t, 0, 0, 1, 1, 0)$ $(0, \pm 1, 0, \pm 1, 0, 0, 0, 0, 0)$
C_{19} contains points of form: velocity vector:	$(q, r, 0, 0, 0, 0, w, w, 0)$ $(0, 0, 0, 0, 0, 0, -1, -1, 0)$

Table 4.1: Location and velocity vectors of the carriers C_0, \dots, C_{19} .

and $C_{17,18}$. The velocity vector of C_{14} is $(+1, 0, -1, 0, 0, 0, 0, 0, 0)$ and the velocity vector of C_{15} is $(-1, 0, +1, 0, 0, 0, 0, 0, 0)$. Accordingly, the velocity vector of C_{17} is $(0, +1, 0, -1, 0, 0, 0, 0, 0)$, while the velocity vector of C_{18} is $(0, -1, 0, +1, 0, 0, 0, 0, 0)$.

Notice that using C_2 for a time interval of positive duration and afterwards changing to C_3 and using C_3 for a time interval of positive duration only leads to an unnecessary detour. Hence, we may for $C_{2,3}$ as well as for $C_{5,6}$, $C_{8,9}$, $C_{11,12}$, $C_{14,15}$ and $C_{17,18}$ assume that the traveller never performs such *unnecessary carrier changes*.

The following definition maps each carrier C_0, \dots, C_{19} to a set of binary vectors of length 5.

$$b(C_i) := \left\{ (u, v, w, x, y) \in \{0, 1\}^5 \mid \begin{array}{l} \exists (q, r, s, t) \in \mathbb{R}^4 \text{ such that} \\ (q, r, s, t, u, v, w, x, y) \text{ belongs to } C_i \end{array} \right\}$$

We obtain that

- $b(C_0) = \{(0, 0, 0, 0, 0)\}$
- $b(C_1) = \{(0, 0, 0, 0, 0), (0, 0, 0, 0, 1)\}$
- $b(C_{2,3}) = \{(0, 0, 0, 0, 1)\}$
- $b(C_4) = \{(0, 0, 0, 0, 1), (0, 0, 0, 1, 0)\}$
- $b(C_{5,6}) = \{(0, 0, 0, 1, 0)\}$
- $b(C_7) = \{(0, 0, 0, 1, 0), (0, 0, 0, 1, 1)\}$
- ...
- $b(C_{17,18}) = \{(0, 0, 1, 1, 0)\}$
- $b(C_{19}) = \{(0, 0, 1, 1, 0), (0, 0, 0, 0, 0)\}$

By construction of the carriers, we obtain

Lemma 4.7. *If $b(C_i) = \{b_1\}$ with $b_1 \in \{0, 1\}^5$, then the affine subspace of the u, v, w, x, y -coordinates of points belonging to C_i is the single point b_1 . If $b(C_i) = \{b_1, b_2\}$ with $b_1, b_2 \in \{0, 1\}^5$, then the affine subspace of the u, v, w, x, y -coordinates of points belonging to C_i is the 1-dimensional line through both points b_1 and b_2 .*

Lemma 4.8. *Let i, j denote two different elements from the set $\{0, \dots, 18\}$. Then $b(C_i) \cap b(C_j) = \emptyset$ implies that the carriers C_i and C_j do not have any point in common.*

Proof. This follows directly from Lemma 4.7, if $|b(C_i)| = 1$ and $|b(C_j)| = 1$. Let us analyze the case that $b(C_i) = \{b_1, b_2\}$ and $b(C_j) = \{b_3\}$. Without loss of generality, we may assume that the binary sequence b_2 is the successor of the binary sequence b_1 . We conclude that $b_2 - b_1$ equals $(0, 0, 0, 0, 1)$, $(0, 0, 0, 1, -1)$, $(0, 0, 1, -1, -1)$, $(0, 1, -1, -1, -1)$ or $(1, -1, -1, -1, -1)$.

The last five coordinates of points belonging to C_i and C_j fulfill:

$$b_1 + \lambda(b_2 - b_1) = b_3 \quad (4.1)$$

We will show that this equation does not have any solution. To this end, let us consider the coordinate ν such that the ν -value of $(b_2 - b_1)$ equals 1. Equation 4.1 restricted to the ν -th coordinate is $(b_1)_\nu + \lambda = (b_3)_\nu$, which implies $\lambda \in \{-1, 0, 1\}$. Because of the assumption that $\{b_1, b_2\} \cap \{b_3\} = \emptyset$, we can exclude that $\lambda = 0$ or $\lambda = 1$ solves Equation 4.1. However, also the case $\lambda = -1$ is not possible for the following reason. Assume that Equation 4.1 is fulfilled for $\lambda = -1$. We know that b_1 as well as $b_2 = b_1 + 1 \cdot (b_2 - b_1)$ are binary vectors, where $b_2 - b_1 \neq 0$. Hence, it is not possible that $b_1 + (-1) \cdot (b_2 - b_1) = b_3$ is also a binary vector. This is a contradiction showing the statement of the Lemma for the case that $b(C_i) = \{b_1, b_2\}$ and $b(C_j) = \{b_3\}$.

The remaining case is that $b(C_i) = \{b_1, b_2\}$ and $b(C_j) = \{b_3, b_4\}$. Again, we assume that b_2 is the binary successor of b_1 and b_4 is the binary successor of b_3 . If $b_2 - b_1 = b_4 - b_3$, then the affine subspaces corresponding to the last five coordinates of C_i and C_j are parallel lines. These lines are disjoint or identical, but the assumption $b(C_i) \cap b(C_j) = \emptyset$ excludes that they are identical.

Let us come to the case that $b_2 - b_1$ and $b_4 - b_3$ are two different vectors from $(0, 0, 0, 0, 1)$, $(0, 0, 0, 1, -1)$, $(0, 0, 1, -1, -1)$, $(0, 1, -1, -1, -1)$ or $(1, -1, -1, -1, -1)$. Without loss of generality, we may assume that $b_2 - b_1$ starts with a smaller subsequence of zeros than $b_4 - b_3$ does. The last five coordinates of points belonging to C_i and C_j fulfill:

$$b_1 + \lambda(b_2 - b_1) = b_3 + \mu(b_4 - b_3) \quad (4.2)$$

We choose the coordinate ν where the ν -value of $b_2 - b_1$ equals 1 and the ν -value of $b_4 - b_3$ equals 0. Equation 4.2 restricted to the ν -th coordinate is $(b_1)_\nu + \lambda = (b_3)_\nu$. Once again, we obtain $\lambda \in \{-1, 0, 1\}$. Restricting Equation 4.2 to the ξ -th coordinate where the ξ -value of $b_2 - b_1$ equals -1 and the ν -value of $b_4 - b_3$ equals 1 provides us the additional information that $\lambda + \mu \in \{-1, 0, 1\}$. All together, only nine combinations for the values of λ and μ are possible. However, all of them contradict the assumption that $\{b_1, b_2\} \cap \{b_3, b_4\} = \emptyset$ as Figure 4.5 shows. \square

The carrier C_{19} has a special role, because this carrier does not fit into the concept of increasing a binary counter. Subspaces not fitting into this concept could cause trouble as the line through $(0, 0)$ and $(1, 1)$ indicates because this line indeed intersects the line through $(0, 1)$ and its binary successor $(1, 0)$. Thus, a more detailed analysis is needed to generalize Lemma 4.8 to the following Lemma.

Lemma 4.9. *Let i, j denote two different elements from the set $\{0, \dots, 19\}$. Then $b(C_i) \cap b(C_j) = \emptyset$ implies that the carriers C_i and C_j do not have any point in common.*

λ	μ	results in	contradicts $\{b_1, b_2\} \cap \{b_3, b_4\} = \emptyset$ because
-1	0	$2b_1 = b_2 + b_3$	$b_1 = b_2 = b_3$
-1	1	$2b_1 = b_2 + b_4$	$b_1 = b_2 = b_4$
-1	2	$2b_1 + b_3 = b_2 + 2b_4$	$b_1 = b_4$ and $b_2 = b_3$
0	-1	$b_1 + b_4 = 2b_3$	$b_1 = b_3 = b_4$
0	0	$b_1 = b_3$	$b_1 = b_3$
0	1	$b_1 = b_4$	$b_1 = b_4$
1	-2	$b_2 + 2b_4 = 3b_3$	$b_2 = b_3 = b_4$
1	-1	$b_2 + b_4 = 2b_3$	$b_2 = b_3 = b_4$
1	0	$b_2 = b_3$	$b_2 = b_3$

Figure 4.5: Equation 4.2 has no solution, if $\lambda \in \{-1, 0, 1\}$ and $\lambda + \mu \in \{-1, 0, 1\}$.

$b_2 - b_1$	b_3	$b_2 - b_1$	b_3	$b_2 - b_1$	b_3	$b_2 - b_1$	b_3	$b_2 - b_1$	b_3
0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	1	0	-1	0
0	1	0	1	1	1	-1	1	-1	1
0	1	1	1	-1	1	-1	1	-1	1
1	0	-1	0	-1	0	-1	0	-1	0

Figure 4.6: For all possible values of $b_2 - b_1$, we can find a coordinate ν such that the ν -value of $b_2 - b_1$ is 1 smaller than the ν -value of $(0, 0, 1, 1, 0)$.

Proof. Thanks to Lemma 4.8, we can restrict ourselves to the case $j = 19$. The affine subspace corresponding to the last five coordinates of carrier points of C_{19} is a line through the origin and the point $b_3 = (0, 0, 1, 1, 0)$. For $2 \leq i \leq 15$, this affine subspace is disjoint from the affine subspaces corresponding to the last five coordinates of points of carriers C_i with $|b(C_i)| = 1$.

Let us analyze the case that $b(C_i) = \{b_1, b_2\}$, where b_2 denotes the binary successor of b_1 . The last five coordinates of points belonging to C_i and C_{19} fulfill $b_1 + \lambda(b_2 - b_1) = \mu \cdot (0, 0, 1, 1, 0)$ for $\lambda, \mu \in \mathbb{R}$. If there is a coordinate ν such that the ν -value of $b_2 - b_1$ equals 0 and the ν -value of $b_3 = (0, 0, 1, 1, 0)$ equals 1, then the above equation restricted to the ν -th coordinate is $(b_1)_\nu = \mu$. In this case, we have $\mu \in \{0, 1\}$. Consequently, the solution candidates for common points of C_i and C_{19} are b_1 and $b_1 + 1 \cdot (b_2 - b_1)$. We conclude that, in this case, no common point of C_i and C_{19} exists by the assumption $b(C_i) \cap b(C_{19}) = \emptyset$. Similar, it can be shown that no solution exists if we can find a coordinate ν such that the ν -value of $b_2 - b_1$ equals -1 and the ν -value of b_3 equals 0.

An enumeration of all possible cases for the difference $b_2 - b_1$ is shown in Figure 4.6. In all cases, it is possible to find a coordinate ν with the desired property.

□

Lemma 4.9 implies that the traveller can not directly change from carrier C_i to carrier C_j , if $b(C_i) \cap b(C_j) = \emptyset$.

Every carrier from C_1 to $C_{17,18}$ in the list of Figure 4.1 has a non-empty intersection with the previous and the subsequent carrier of the list. There are also intersections of another type which we call *redundant intersections*. Such a redundant intersection occurs between C_1 and C_4 . The intersection of the carriers C_1 and C_4 is a subset of the carrier $C_{2,3}$. This means that, if a direct carrier change from C_1 to C_4 should be possible at all, we could virtually replace this change by two simultaneous changes, one from C_1 to $C_{2,3}$ and one from $C_{2,3}$ to C_4 . Of course, our model does not allow to perform two carrier changes at the same time point. It is just a notation for the analysis that we say that the traveller changes from C_1 to $C_{2,3}$ and at the same moment from $C_{2,3}$ to C_4 . This notation is convenient to avoid case differentiations. Furthermore, with this notation, no direct change from C_1 to C_4 is necessary, and that is the reason why the intersection of C_1 and C_4 is called redundant.

The intersection graph of all carriers is shown in Figure 4.7.

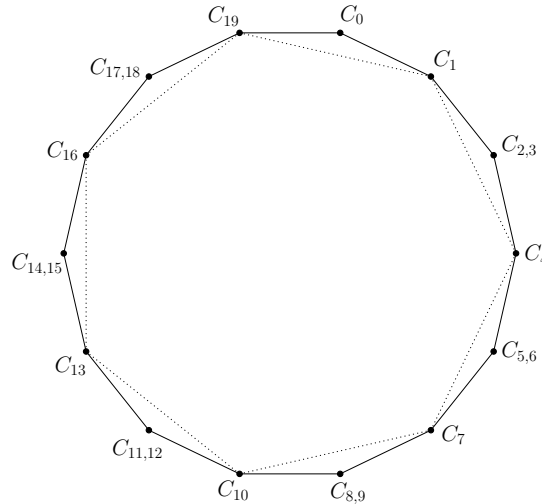


Figure 4.7: Intersection graph of the carriers C_0, \dots, C_{19} . Dotted edges correspond to redundant intersections.

For our analysis, we would like to talk about the *first* carrier change, the *second* carrier change, the *third* carrier change and so on. However, this terminology might not apply to the case of traveller paths having an infinite number of carrier changes.

The following Lemma ensures that, in case of the carriers C_0, \dots, C_{19} , we do not have to take care about infinitely many carrier changes. The Lemma is restricted to traveller paths not performing unnecessary carrier changes; see page 57 for the definition of unnecessary carrier changes.

Lemma 4.10. *In case of the instance C_0, \dots, C_{19} , every traveller path $([0, T], f, A, g)$ not performing unnecessary carrier changes has only a finite number of carrier changes.*

Proof. Let κ denote the minimum distance of any two non-intersecting carriers from C_0, \dots, C_{19} . We assume, for contradiction, that $|A| = \infty$. Thanks to the infinite pigeon hole principle, we can choose an interval $I' \subset [0, T]$ of size $< \frac{\kappa}{\sqrt{3}}$ such that $|A \cap I'| = \infty$. By Lemma 4.5, there exists an interval $I \subset I'$ such that $6 \leq |A \cap I| < \infty$. It makes sense to talk about the *first* carrier change in I , the *second* carrier change, the *third* one and so on.

Since the size of I is smaller than $\frac{\kappa}{\sqrt{3}}$ and the maximum carrier speed is $\sqrt{3}$, we conclude that the traveller does not visit any two non-adjacent vertices of the graph shown in Figure 4.7 during the time interval I . He only visits vertices of one clique. If the traveller visits only two carriers, the contradiction is that, by construction of the carriers, at least one of both carriers moves him away from the intersection with the other carrier.

Let us come to the case that the traveller, during the time interval I , visits all 3 vertices of a clique of the graph shown in Figure 4.7. For the moment, we assume that this clique does not contain the vertex C_{19} . Thus, the clique has two vertices such that the corresponding carriers monotonously increase the $4w + 2x + y$ -value of the traveller. The third carrier $C_{i,i+1}$ of the clique has the property that $|b(C_{i,i+1})| = 1$. Consequently, all points of this carrier have the same $4w + 2x + y$ -value. If the traveller visits $C_{i,i+1}$ and afterwards uses another carrier of the clique for a positive time duration, then it is not possible for him to return to $C_{i,i+1}$. Hence, we are in the above case of only two carriers after the traveller visited $C_{i,i+1}$ for the first time.

The remaining issue is to show that it is not possible to change between the carriers C_0, C_1 and C_{19} infinitely often during the time interval I . However, if the traveller changes from C_0 or C_1 to C_{19} , then his w -coordinate decreases below 0 and no further carrier change is possible afterwards. \square

Suppose, the traveller is located on the base carrier C_0 and wants to use the other carriers on a trip which finally brings him back to C_0 . If the traveller starts his trip with a carrier change from C_0 to C_{19} , then his w -coordinate will decrease below 0 and no other carrier change is possible afterwards. Hence, all traveller paths from a point in C_0 to a different point in C_0 have their first carrier change from C_0 to C_1 .

By construction of the carriers, whenever the traveller changes between two of them, his new carrier moves him away from the common intersection of both carriers. It is not possible to change back to the previous carrier directly.

Consequently, the intersection behavior of the carriers shown in Figure 4.7 leads to the following Lemma.

Lemma 4.11. *The only possibility for a non-constant path from C_0 to C_0 without any intermediate stops in C_0 is to use all carriers $C_0, C_1, C_{2,3}, C_4, \dots, C_{19}, C_0$ in exactly*

$$\begin{array}{l}
\begin{array}{c} C_1 \\ \xrightarrow{\quad} \end{array} \begin{pmatrix} e, & f, & 0, & 0, & 0, & 0, & 0, & 0, & 0 \end{pmatrix} \\
\begin{array}{c} C_{2,3} \\ \xrightarrow{\quad} \end{array} \begin{pmatrix} e, & f, & 0, & 0, & 0, & 0, & 0, & 0, & 1 \end{pmatrix} \\
\begin{array}{c} C_4 \\ \xrightarrow{\quad} \end{array} \begin{pmatrix} e, & f, & e', & 0, & 0, & 0, & 0, & 0, & 1 \end{pmatrix} \\
\begin{array}{c} C_{5,6} \\ \xrightarrow{\quad} \end{array} \begin{pmatrix} e, & f, & e', & f', & 0, & 0, & 0, & 1, & 0 \end{pmatrix} \\
\begin{array}{c} C_7 \\ \xrightarrow{\quad} \end{array} \begin{pmatrix} e, & f, & e', & f', & 0, & 0, & 0, & 1, & 1 \end{pmatrix} \\
\begin{array}{c} C_{8,9} \\ \xrightarrow{\quad} \end{array} \begin{pmatrix} 0, & f, & e', & f', & 0, & 0, & 0, & 1, & 1 \end{pmatrix} \\
\begin{array}{c} C_{10} \\ \xrightarrow{\quad} \end{array} \begin{pmatrix} 0, & f, & e', & f', & 0, & 0, & 1, & 0, & 0 \end{pmatrix} \\
\begin{array}{c} C_{11,12} \\ \xrightarrow{\quad} \end{array} \begin{pmatrix} 0, & 0, & e', & f', & 0, & 0, & 1, & 0, & 0 \end{pmatrix} \\
\begin{array}{c} C_{13} \\ \xrightarrow{\quad} \end{array} \begin{pmatrix} 0, & 0, & e', & f', & 0, & 0, & 1, & 0, & 1 \end{pmatrix} \\
\begin{array}{c} C_{14,15} \\ \xrightarrow{\quad} \end{array} \begin{pmatrix} e', & 0, & 0, & f', & 0, & 0, & 1, & 0, & 1 \end{pmatrix} \\
\begin{array}{c} C_{16} \\ \xrightarrow{\quad} \end{array} \begin{pmatrix} e', & 0, & 0, & f', & 0, & 0, & 1, & 1, & 0 \end{pmatrix} \\
\begin{array}{c} C_{17,18} \\ \xrightarrow{\quad} \end{array} \begin{pmatrix} e', & f', & 0, & 0, & 0, & 0, & 1, & 1, & 0 \end{pmatrix} \\
\begin{array}{c} C_{19} \\ \xrightarrow{\quad} \end{array} \begin{pmatrix} e', & f', & 0, & 0, & 0, & 0, & 0, & 0, & 0 \end{pmatrix}
\end{array}$$

Figure 4.8: The traveller, located at $(e, f, 0, 0, 0, 0, 0, 0, 0, 0)$, uses all carriers $C_0, C_1, C_{2,3}, C_4, \dots, C_{19}, C_0$ in exactly this order.

this order.

Let us analyze what happens to the traveller located at the point $(e, f, 0, 0, 0, 0, 0, 0, 0, 0)$ in C_0 , if all carriers are used in this order. We define $e' := \alpha e + \beta f + \gamma$ and $f' := \lambda e + \mu f + \nu$. The resulting trip is shown in Figure 4.8.

Lemma 4.12. *Using the carriers $C_0, C_1, \dots, C_{19}, C_0$ has the same effect on the traveller's location as the affine mapping h .*

Of course, after using the carriers C_0, C_1, \dots, C_{19} and having returned to C_0 , the traveller may feel free to take the tour again which corresponds to iterating the affine mapping h .

The generalization to the case that we are dealing with 5 affine mappings instead of one is straightforward. For the second affine mapping, we construct carriers C_{20}, \dots, C_{38} . The last five coordinates of these carriers are chosen such that

$$\begin{aligned}
b(C_{20}) &= \{(0, 0, 0, 0, 0), (0, 0, 1, 1, 1)\}, \\
b(C_{21}) &= \{(0, 0, 1, 1, 1)\}, \\
&\dots \\
b(C_{38}) &= \{(0, 1, 1, 0, 0), (0, 0, 0, 0, 0)\}.
\end{aligned}$$

Accordingly, the carriers for the third, fourth and fifth affine mapping are created. The final carrier set is $C = \{C_0, C_1, \dots, C_{95}\}$.

Lemma 4.13. *Let i, j denote two different elements from the set $\{0, \dots, 95\}$. Then $b(C_i) \cap b(C_j) = \emptyset$ implies that the carriers C_i and C_j do not have any point in common.*

Proof. Let $C^* := \{C_{19}, C_{38}, C_{57}, C_{76}, C_{95}\}$ denote the set of carriers not fitting into the concept of increasing a binary counter.

The case that C_i and C_j belong to $C \setminus C^*$ is captured by the proof of Lemma 4.8.

If C_i and C_j belong to C^* , then the origin 0 belongs to C_i as well as to C_j , which means that the precondition of Lemma 4.13 is not fulfilled and we have nothing to show.

Let us come to the case that $C_i \in C \setminus C^*$ and $C_j \in C^*$. If $|b(C_i)| = 1$, then the statement of the Lemma is obvious. Otherwise, the Lemma can be proved by a complete enumeration of all cases as it was done for the affine mapping h in Figure 4.6. Let b_1 and b_2 denote the points belonging to $b(C_i)$ such that b_2 is the binary successor of b_1 . Hence, $b_2 - b_1$ is $(0, 0, 0, 0, 1)$, $(0, 0, 0, 1, -1)$, $(0, 0, 1, -1, -1)$, $(0, 1, -1, -1, -1)$ or $(1, -1, -1, -1, -1)$.

Let b_{38}, b_{57}, b_{76} and b_{95} denote the unique point in $b(C_{38}), b(C_{57}), b(C_{76}), b(C_{95})$ different from 0. For each b_{19k} , we can finish the proof as it was done in the proof of Lemma 4.9 by showing that, for all possible values of $b_2 - b_1$, we can find a coordinate ν such that the ν -value of $b_2 - b_1$ is 1 smaller than the ν -value of b_{19k} . This is always possible as Figure 4.9 shows.

□

By Lemma 4.13, carriers belonging to different affine mappings intersect only in the base carrier C_0 . The answer to FIVE-AFFINE-MAPPINGS-REACHABILITY is “yes” if and only if there exists an s -to- g traveller path in the corresponding instance of Traveller’s Problem.

We obtain

Lemma 4.14. *Traveller’s Problem in 9-dimensional space is undecidable, even if the number of carriers is restricted to 96.*

For our carrier construction with the carriers C_0, \dots, C_{96} to prove Lemma 4.14, we used two dimensions (namely s and t) to buffer the function values of the 1-dimensional affine mappings. However, it is easy to see that one dimension to buffer such values is enough:

Suppose, we want to compute the value of the affine mapping:

$$h : (q, r) \mapsto (\alpha q + \beta r + \gamma, \lambda q + \mu r + \nu) =: (h_1(q, r), h_2(q, r))$$

If $\alpha \neq 0$, this can be done in the following way. First, we compute $q' := h_1(q, r)$ by moving carriers using dimension s as buffer and store the outcome q' in the q -

$b_2 - b_1$	b_{38}	$b_2 - b_1$	b_{38}	$b_2 - b_1$	b_{38}	$b_2 - b_1$	b_{38}	$b_2 - b_1$	b_{38}
0	0	0	0	0	0	0	0	1	0
0	1	0	1	0	1	1	1	-1	1
0	1	0	1	1	1	-1	1	-1	1
0	0	1	0	-1	0	-1	0	-1	0
1	0	-1	0	-1	0	-1	0	-1	0

$b_2 - b_1$	b_{57}	$b_2 - b_1$	b_{57}	$b_2 - b_1$	b_{57}	$b_2 - b_1$	b_{57}	$b_2 - b_1$	b_{57}
0	1	0	1	0	1	0	1	1	1
0	0	0	0	0	0	1	0	-1	0
0	0	0	0	1	0	-1	0	-1	0
0	1	1	1	-1	1	-1	1	-1	1
1	0	-1	0	-1	0	-1	0	-1	0

$b_2 - b_1$	b_{76}	$b_2 - b_1$	b_{76}	$b_2 - b_1$	b_{76}	$b_2 - b_1$	b_{76}	$b_2 - b_1$	b_{76}
0	1	0	1	0	1	0	1	1	1
0	1	0	1	0	1	1	1	-1	1
0	0	0	0	1	0	-1	0	-1	0
0	0	1	0	-1	0	-1	0	-1	0
1	0	-1	0	-1	0	-1	0	-1	0

$b_2 - b_1$	b_{95}	$b_2 - b_1$	b_{95}	$b_2 - b_1$	b_{95}	$b_2 - b_1$	b_{95}	$b_2 - b_1$	b_{95}
0	1	0	1	0	1	0	1	1	1
0	1	0	1	0	1	1	1	-1	1
0	1	0	1	1	1	-1	1	-1	1
0	1	1	1	-1	1	-1	1	-1	1
1	0	-1	0	-1	0	-1	0	-1	0

Figure 4.9: For all possible values of $b_2 - b_1$ and b_{19k} , we can find a coordinate ν such that the ν -value of $b_2 - b_1$ is 1 smaller than the ν -value of b_{19k} .

coordinate. Afterwards, we determine the function value of the affine mapping

$$h'_2 : (q, r) \mapsto \frac{\lambda}{\alpha}q + \left(\mu - \frac{\beta\lambda}{\alpha}\right)r + \nu - \frac{\gamma\lambda}{\alpha}$$

again using dimension s as buffer. Notice that $h'_2(q', r) = h_2(q, r)$.

If $\alpha = 0$ and $\lambda \neq 0$, we can simply reverse the roles of $\alpha q + \beta r + \gamma$ and $\lambda q + \mu r + \nu$.

If $\alpha = 0$ and $\lambda = 0$, we have an even simpler case where the q -coordinate is completely irrelevant for the affine mapping h .

Let us assume that $\alpha \neq 0$ and describe the changes which are necessary to compute the affine mapping h in the above manner. We use only dimension s to buffer function values. Dimension t is no longer needed.

No changes are done with respect to the last five coordinates of the carriers C_0, \dots, C_{19} . This ensures that the results about the intersection behavior of the carriers remain unaffected. The carriers C_0, \dots, C_4 keep as they are. The carrier $C_{5,6}$ is now used to bring the traveller's q -coordinate to 0. The carrier $C_{8,9}$ is now defined such that the traveller can use it to swap the values of his q and his s -coordinate. The carrier $C_{11,12}$ is used to compute the function value of h'_2 with respect to the current value of the q -coordinate. The result equals the value of the original mapping h_2 applied to the traveller's initial qr -coordinates. Again, the result is stored in coordinate s . Carrier $C_{14,15}$ brings the traveller's r -coordinate to 0. And, finally, carrier $C_{17,18}$ swaps the values of the s and the r -coordinate.

This construction saves one dimension. We conclude

Theorem 4.15. *Traveller's Problem with innate speed zero is undecidable in dimension $d \geq 8$.*

4.4 Dimension two

Clearly, Traveller's Problem is most interesting in dimension 2 and 3. In this section we provide some results for the two-dimensional case.

4.4.1 NP-hardness

The first result shows that Traveller's Problem in the plane is NP-hard, and does not even admit a constant factor approximation, unless $P=NP$, even if all carriers are vertical line segments. Our reduction is from the NP-complete problem PARTITION [42].

PARTITION: Given n natural numbers a_1, a_2, \dots, a_n , let $S := \sum_{i=1}^n a_i$. Does there exist a subset $I \subset \{1, \dots, n\}$ such that $\frac{S}{2} = \sum_{i \in I} a_i$?

Theorem 4.16. *Traveller's Problem in dimension two is NP-hard for arbitrary innate speed. There is no constant factor approximation, unless $P=NP$.*

Proof. Let us first assume that the innate speed v equals 0. Our construction uses vertical carriers C, C' and $A_i, B_i, 1 \leq i \leq n$, of height S each. C and C' are aligned and do not move. The bottommost point of C equals s , while g is the center point of C ; see Figure 4.10. For each i , carrier A_i moves in such a way that, at some time, it is congruent with C , and has gained a_i in height on reaching C' afterwards. B_i traverses the horizontal strip containing C, C' from right to left. These carriers pass over C, C' in the order $A_1, B_1, A_2, B_2, \dots, A_n, B_n$, with plenty of time in between so that no interference is possible.

The traveller, located at some point on C , may choose to board a carrier A_i . It will get him to C' , at a point by a_i higher than his point of departure from C . Then, he can use B_i , or one of the later carriers B_j , to return to C while maintaining his height. In other words, for each index i , the traveller has the option to move a distance a_i upwards on C . Thus, he is able to reach its middle point, g if, and only if, a partition of the given numbers is possible.

Now assume that a partition is impossible. Then the traveller misses g by a distance at least $\frac{1}{2}$, since all a_i are natural numbers. If innate speed $v = 0$, the traveller will never get to g . Now let $v > 0$, and assume that there exists an approximation algorithm with approximation factor $< \alpha$. By speeding up the carriers we can ensure that $t' = \frac{1}{2\alpha v}$ holds for the time t' where B_n hits C . At this time point, the traveller has walked a total distance of at most $t'v = \frac{1}{2\alpha}$, so that he is at least $d := \frac{1}{2} - \frac{1}{2\alpha}$ away from g , if he has reached C at all. To walk distance d takes time at least $\frac{d}{v} = d2\alpha t' = (\alpha - 1)t'$, so that the whole journey needs time $\alpha t'$ at least. Thus, the approximation algorithm would decide PARTITION. \square

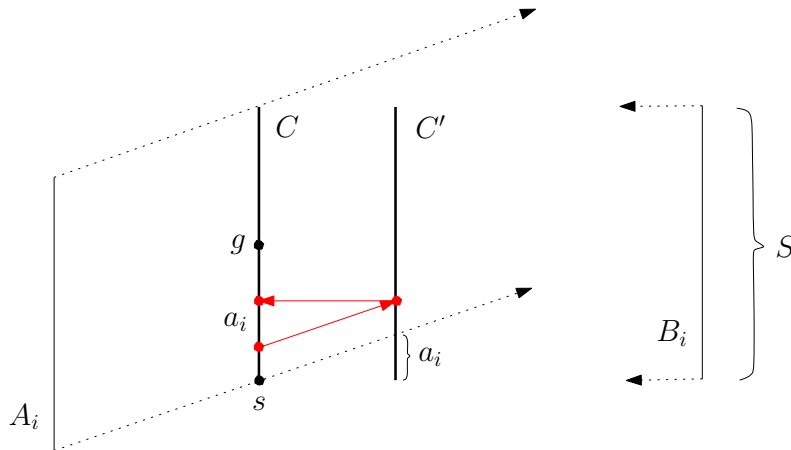


Figure 4.10: Proving Traveller's Problem NP-hard.

A small perturbation of the carriers in the above construction shows that the problem remains NP-hard also, if no parallel carriers are allowed.

4.4.2 Decidability

The challenge is to upper bound the complexity of Traveller's Problem in dimension 2. Straightforward attempts meet with firm resistance because of the difficulty of instances requiring an infinite number of carrier changes; see the first example of Section 4.1.2 on page 49.

4.4.2.1 Analyzing paths by means of the graph G

We will establish that the two-dimensional Traveller's Problem is decidable for *restricted* instances.

Definition 4.17. *A two-dimensional Traveller's Problem instance is restricted, if:*

- *the shape of each carrier is given by a point, line segment, half-line or line,*
- *no two carriers of positive length have parallel orientation, and*
- *the traveller's innate speed is zero.*

For restricted instances, any two carriers intersect in at most one point at any time.

We start our analysis with capturing the combinatorial structure of traveller paths with a finite number of carrier changes by mapping them to a graph $G = (V(G), E(G))$.

Definition 4.18. *The graph G contains one vertex for each ordered pair of carriers.*

$$V(G) = \{(c, d) : c, d \in C, c \neq d\}$$

A directed edge goes from vertex (c, d) to vertex (r, u) , if $d = r$.

$$E(G) = \{((c, d), (d, u)) : c, d, u \in C, c \neq d, d \neq u\}$$

Definition 4.19. *If $(I, f, \{a_1, a_2, \dots, a_m\}, g)$ denotes a traveller path such that $\min I < a_1 < a_2 < \dots < a_m < \max I$, then the map ψ from $A = \{a_1, a_2, \dots, a_m\}$ to the graph vertices $V(G)$ is defined as:*

$$\psi : A \rightarrow V(G)$$

$$a_i \mapsto \begin{cases} (g((\min I, a_1)), g((a_1, a_2))), & \text{for } i = 1 \\ (g((a_{i-1}, a_i)), g((a_i, a_{i+1}))), & \text{for } 2 \leq i \leq m - 1 \\ (g((a_{m-1}, a_m)), g((a_m, \max I))), & \text{for } i = m \end{cases}$$

For $i \in \{1, \dots, m-1\}$, the second component of $\psi(a_i)$ equals the first component of $\psi(a_{i+1})$. Hence, $(\psi(a_i), \psi(a_{i+1})) \in E(G)$. We conclude that, for $|A| < \infty$, the function ψ maps the traveller path to a path in the graph G .

Definition 4.20. For $m \geq 1$, the traveller path $(I, f, \{a_1, \dots, a_m\}, g)$ with $\min I < a_1 < \dots < a_m < \max I$ is called a cycle, if

- for c defined as the first component of $\psi(a_1)$ and d as the second component of $\psi(a_m)$, the intersection of c and d equals $f(\min I)$ at time $\min I$ and equals $f(\max I)$ at time $\max I$, and
- the path

$$\chi := ((d, c), \psi(a_1), \dots, \psi(a_m), (d, c))$$

is a cycle in G , which means that the vertices $\psi(a_1), \dots, \psi(a_m), (d, c)$ are pairwise different.

The travel time of the cycle is defined as $\max I - \min I$. We say that the traveller path $(I, f, \{a_1, \dots, a_m\}, g)$ is a realization of the cycle χ in G .

Definition 4.21. In general, and no longer assuming that $|A| < \infty$, the traveller path (I, f, A, g) is said to contain a cycle if there exists an interval $I' \subset I$ such that $1 \leq |A \cap I'| < \infty$ holds and the traveller path restricted to I' fulfills Definition 4.20.

Lemma 4.22. Any traveller path (I, f, A, g) with $|A| > n(n-1)$ contains a cycle.

Proof. If $|A| < \infty$, we define $\tilde{I} := I$. Otherwise, Lemma 4.5 guarantees that we can choose an interval $\tilde{I} \subset I$ with $n(n-1) < |A \cap \tilde{I}| < \infty$. Now, we consider the path (I, f, A, g) restricted to \tilde{I} . Let $a_1 < \dots < a_m$ denote the elements of $A \cap \tilde{I}$. We define ϕ as the path $(\psi(a_1), \dots, \psi(a_m))$ in G . Since $m = |A \cap \tilde{I}|$, which is the number of vertices of ϕ , is greater than $|V(G)| = n(n-1)$, we conclude that ϕ visits at least one vertex of $V(G)$ twice. Hence, ϕ contains a cycle. \square

Let $\Gamma(n)$ denote the finite number of possible cycles in G . If $|A| > (k\Gamma(n) + 1)n(n-1)$ for $k \in \mathbb{N}$, then, by the pigeonhole principle and Lemma 4.22, at least one cycle of G is realized at least $k+1$ times by (I, f, A, g) . In particular, we obtain the following result for $|A| = \infty$.

Lemma 4.23. Any traveller path (I, f, A, g) with $|A| = \infty$ realizes at least one cycle of G infinitely often.

In the following we will assign time-dependent *edge weights* to the edges of G . For any edge $e \in E(G)$, its edge weight $w_t(e)$ at time t is either a real number, or an interval, or ∞ . Roughly spoken, edge $e = ((c, d), (d, u))$ should have edge weight $w_t(e)$, if the traveller, located at the intersection point of c and d at time t , is able to use carrier d

for duration $w_t(e)$ and this brings him to the intersection of d and u . It will turn out soon that for any edge e , the function $t \mapsto w_t(e)$ has a very simple structure.

The set $F(c, d) := \{t \in \mathbb{R}_{\geq 0} : c \text{ and } d \text{ have non-empty intersection at time } t\}$ is a closed interval, which may be unbounded to the right side. For $t \in F(c, d)$, let $s_t(c, d)$ denote the intersection point of c and d at time t . Assume that the traveller is located at $s_t(c, d)$ at time t . From there on using only carrier d brings him to location $J(t') := s_t(c, d) + (t' - t)\vec{d}$ at time t' , where \vec{d} denotes the velocity vector of d . Thanks to the linear movement of the carriers, the intersection point $K(t') := s_{t'}(d, u)$ of d and u moves linearly for growing $t' \in R$, where $R := \mathbb{R}_{\geq t} \cap F(d, u)$. Since $J(t')$ and $K(t')$ move linearly, only the following three cases are possible.

Definition 4.24. *Definition of the edge weight $w_t(e)$:*

(i) $J(t') \neq K(t') \forall t' \in R$

We assign $w_t(e) = \infty$ as edge weight.

(ii) There is exactly one $t' \in R$ solving $J(t') = K(t')$.

We assign $w_t(e) = t' - t$.

(iii) $J(t') = K(t') \forall t' \in R$, where $\min R < \max R$

We assign the complete interval, $w_t(e) = [\min R - t, \max R - t]$.

In case (i), carrier d does not bring the traveller to carrier u . In case (ii), and if $t' > t$, there is a traveller path which starts at $s_t(c, d)$ at time t , uses carrier d and ends at the intersection of d and u at time $t + w_t(e)$. In case (iii), the traveller is able to follow the intersection $s_{t'}(d, u)$ by using carrier d . An example of case (iii) is shown in Figure 4.12 where carrier c_2 enables the traveller to stay in the intersection of c_2 and c_3 .

Definition 4.25. *Edge e is called feasible at time t , if $w_t(e) \neq 0$ and $w_t(e) \neq \infty$.*

Lemma 4.26. *For any edge $e = ((c, d), (d, u))$, the set of time points when e is feasible is an interval which can be computed in time $O(1)$.*

Proof. Edge e is feasible at time t , if and only if there exists a $t' \in \mathbb{R}_{>t} \cap F(d, u)$ such that

$$s_t(c, d) + (t' - t)\vec{d} = s_{t'}(d, u)$$

This equation involves two-dimensional vectors on both sides. It can be decomposed into two linear equations. Both of them have two variables, namely t and t' . Solving this linear equation system leads to an affine solution subspace Q . Intersecting Q with the convex set $\{(t, t') \in \mathbb{R}^2 : t' \in \mathbb{R}_{>t} \cap F(d, u)\}$ and projecting the result on the t -axis gives us the interval of feasible time points of e . \square

The argument of the proof above also gives us the following information.

Lemma 4.27. *Let $e \in E(G)$ denote any edge whose set of feasible time points is not empty. Then, one of the following two cases arises:*

case (iii): *During the feasibility of e , its weight is a (time-dependent) interval of positive size, or*

case (ii): *during the feasibility of e , its weight is a (time-dependent) real number. In this case, the edge weight is a linear function of the time.*

Thanks to the above Lemma, it makes sense to differentiate between *case (ii) edges* and *case (iii) edges*.

In the following we assume that the carrier set C contains a point-shaped *goal carrier* whose location matches the goal all the time. Accordingly, for every carrier c containing the start location at time 0, we assume that there is also a point-shaped carrier c' matching s at time 0 and having the same velocity vector as c .

A direct consequence of Lemma 4.26 and 4.27 is the following.

Lemma 4.28. *In time $O(n^3)$, we can compute a time bound ω such that the set of feasible edges of G does not change after time ω , and $w_t(e) = [0, \infty] \forall t \geq \omega$ for all case (iii) edges.*

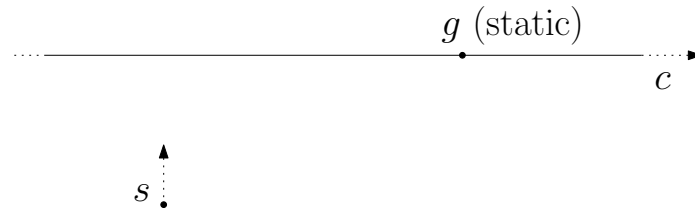


Figure 4.11: Time ω is when the start carrier s intersects the line-shaped carrier c . The traveller can reach the goal g , but not until time ω .

As Figure 4.11 shows, even if the traveller can reach the goal, he cannot necessarily be there until time ω . After time ω , the connected components of G with respect to the feasible edges do not change. This has the following consequence. If a traveller path corresponds to a path χ in G which contains cycles after time ω , then there is also a path χ' in G which has no cycles, uses only edges feasible after time ω and ends up at the same vertex as χ . Clearly, the arrival times at the vertices of G may be different for χ and χ' . However, the crucial observation is that, if the goal location can be reached at all, it can be done without any cycles after time ω . Traveller paths to the goal location correspond to paths in G ending up at a vertex of G whose second component equals the goal carrier.

If $e = ((c, d), (d, u))$ denotes a case (iii) edge, then the reachability of the goal is not affected if we replace any use of e after time ω by a direct change from c to u . Hence,

for upper bounding the travel time needed to reach the goal, we may assume that the traveller uses only case (ii) edges after time ω . Then, the first vertex visited by the traveller after time ω is reached until time $\hat{\omega} := \omega + W$, where W denotes the maximum case (ii) edge weight attained in $[0, \omega]$.

For $t > \omega$, we define

$$\tau(t) = \max\{w_{\hat{t}}(e) : \omega \leq \hat{t} \leq t \text{ and } e \text{ is case (ii) edge after time } \omega\}$$

The second vertex visited after time ω is visited until time $\hat{\omega} + \tau(\hat{\omega})$. For $\mu(t) := t + \tau(t)$, the k -th vertex after time ω is visited until time $\mu^{(k-1)}(\hat{\omega})$. Suppose, an s -to- g traveller path exists. Since the number of vertices which has to be visited after time ω to reach the goal is less than n^2 , the minimum arrival time can be upper bounded by $\Omega := \mu^{(n^2)}(\hat{\omega})$.

Theorem 4.29. *For any restricted instance of Traveller's Problem, we can compute a time bound Ω such that, if it is possible to reach the goal location at all, then it is also possible until time Ω .*

4.4.2.2 Propagation of reachable points

Definition 4.30. *A point $l \in \mathbb{R}^2$ is called reachable at time t , if there exists a traveller path $([0, t], f, A, g)$, such that $f(0)$ equals the traveller's start location s and $f(t) = l$.*

How does the set of reachable points on all carriers develop when time increases? At time $t = 0$, only the start location is reachable. If we assume that the start location belongs to only one carrier, the traveller's trip is uniquely determined until for the first time another carrier intersects the reachable point. Then the traveller may stay on his current carrier or change to the other one. After this moment we have reachable points on both carriers. As Figure 4.12 shows, it is even possible that complete reachable intervals arise. A single point is also considered an interval for now.

In the following we use *propagation phases* to describe the situation that reachability is passed on by a reachable interval on one carrier gliding over parts of another carrier.

Definition 4.31. *A propagation phase consists of the following data:*

- a reachable interval $R(t)$ on a carrier c ,
- a carrier c' different from c , and
- the closed time interval $J \neq \emptyset$ where $R(t)$ and c' have non-empty intersection.

The reachable interval $R(t)$ is called the *source* of the propagation phase and the carrier c' the *target carrier* of the propagation phase. We denote the time point $\min J$

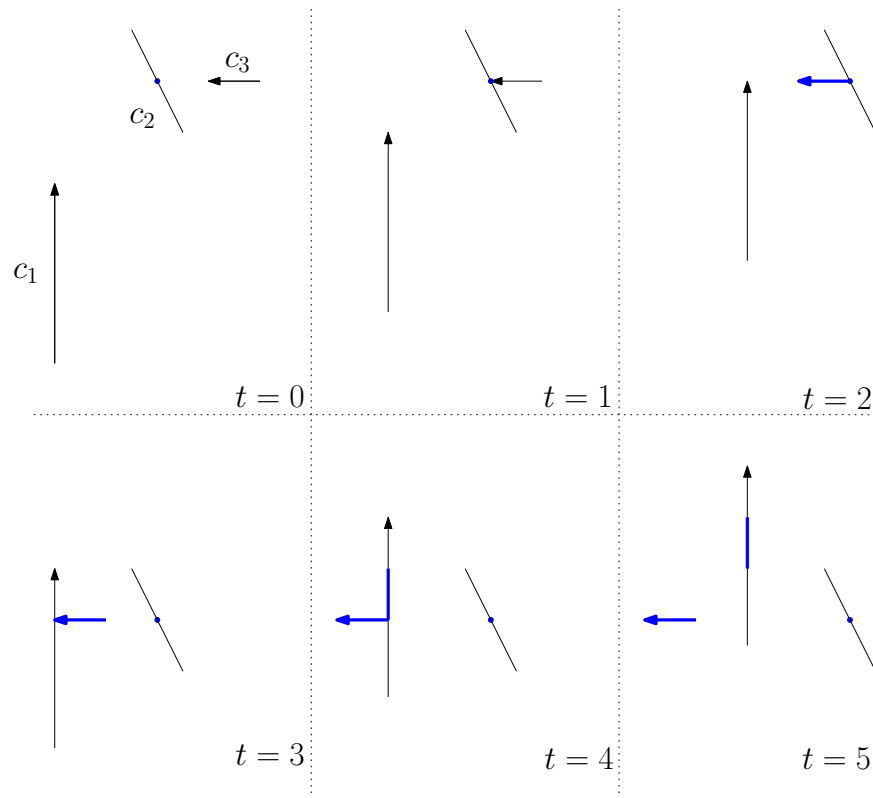


Figure 4.12: Development of reachable intervals which are marked as blue. Snapshots are taken at time $t = 0, 1, \dots, 5$. Carrier c_1 is moving upwards with speed 1, carrier c_2 is static, carrier c_3 is moving to the left with speed 1.

as the start of the propagation phase and the time point $\max J$ as the end of the propagation phase.

The bounds of the reachable intervals and the bounds of the carriers have to be taken into account to determine the start and the end of the propagation phases.

Different propagation phases can happen simultaneously. It may also happen that a propagation phase which has not yet reached its end already caused an interval producing a new propagation phase. This can even go so far as to lead to the situation shown in Figure 4.13 where a reachable interval expands itself via other carriers. The

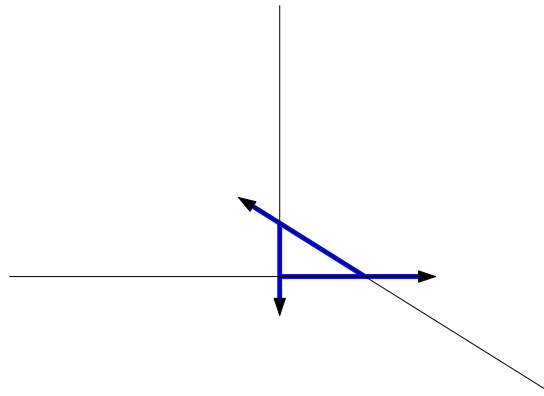


Figure 4.13: We have to prevent these reachable intervals from “feeding themselves” in the future.

instances of Figure 4.2 and Figure 4.13 indicate that cycles, and in particular cycles of small travel time, are an important challenge. For the moment, we exclude arbitrary small cycles by restricting ourselves to κ -safe instances.

Definition 4.32. For $\kappa > 0$, a restricted (definition 4.17 on page 67) instance of Traveller’s Problem is κ -safe, if all cycles of traveller paths (starting from an arbitrary location at an arbitrary time) have travel time $> \kappa$.

The following Lemma helps us to avoid circular self-dependencies as shown in Figure 4.13.

Lemma 4.33. For any κ -safe instance, no propagation phase contributes to extending its own duration within a time interval $[t, t']$ of size at most κ .

Proof. If a propagation phase from a reachable interval on carrier c to carrier d contributes to extending its own duration in $[t, t']$, then there exists a traveller path using the carrier sequence c, d, \dots, c, d in $[t, t']$. But such a traveller path contains a cycle, and thus needs travel time at least κ . \square

Lemma 4.33 makes it easy to avoid circular self-dependencies of the propagation phases in case of κ -safe instances. For all $t \in \{0, \kappa, 2\kappa, \dots\}$, we finalize all reachable intervals. Afterwards, they are not changed anymore and all propagation phases which are still active are continued with new target intervals.

Our propagation algorithm has to process the discrete time points when a propagation phase starts or ends. Due to the linear movement of the carriers, it is easy to determine the next such event for each reachable interval. The events are handled in the order of the corresponding time points.

If the algorithm finally creates a reachable interval containing the goal location, we are done. On the other hand, if the goal does not belong to a reachable interval at the time bound Ω of Theorem 4.29, then no traveller path to the goal location exists.

The remaining issue is to show that the propagation effort until time Ω is finite.

Definition 4.34. *The propagation graph contains a vertex for each propagation phase and an additional root vertex. There is a directed edge from propagation phase P to propagation phase Q , if Q is induced by a reachable interval which was created by P . All propagation phases induced by the reachable interval corresponding to the traveller's start location are connected to the root vertex.*

For example, the propagation graph corresponding to the propagation history shown in Figure 4.12 is just a directed path of length 2. Lemma 4.33 and the finalizing of all reachable intervals at time $t = 0, \kappa, 2\kappa, \dots$ ensure that the propagation graph does not contain any directed cycles. Since each node (except for the root) is incident to exactly one incoming edge, we conclude that the propagation graph also does not contain undirect cycles. Hence, it is a tree. Any path starting from the root of the propagation graph can be realized by a corresponding traveller path.

Lemma 4.35. *For a κ -safe instance, any path starting from the root of the propagation graph has length at most $\lceil \frac{\Omega}{\kappa} \rceil n^2$.*

Proof. If there is a directed path in the propagation graph with length $> \lceil \frac{\Omega}{\kappa} \rceil n^2$, then there is also a traveller path (I, f, A, g) with $|A| > \lceil \frac{\Omega}{\kappa} \rceil n^2$. We partition the interval $[0, \Omega]$ into $\lceil \frac{\Omega}{\kappa} \rceil$ intervals of equal size. Each resulting interval has length at most κ . By the pigeonhole principle, at least one of them contains $> n^2$ points of A . By Lemma 4.22, the traveller path contains a cycle of travel time at most κ in this interval which contradicts the precondition that the instance is κ -safe. \square

Lemma 4.36. *For κ -safe instances the propagation graph is finite.*

Proof. Within each time interval $[j\kappa, (j+1)\kappa]$, each reachable interval on any carrier c induces at most one propagation phase with respect to each carrier different from c . We conclude that all vertices of the propagation graph have out-degree at most $n \lceil \frac{\Omega}{\kappa} \rceil$. The in-degree of all vertices (except for the root) is 1. The propagation graph is a tree whose vertices have finite degree. If such a tree contained infinitely many vertices,

there had to be at least one infinite path by König's infinity Lemma [44]. However, such a path cannot exist because of Lemma 4.35. \square

We conclude that the propagation effort until time Ω is finite and obtain the following Theorem.

Theorem 4.37. *Traveller's Problem for κ -safe instances is decidable.*

Instead of computing a suitable value for κ in advance, the algorithm can also be performed such that whenever a propagation would start to close a cycle in the propagation graph, all reachable intervals are finalized and all active propagation phases are continued with new target intervals.

4.4.2.3 Treatment of Zeno cycles

Let us consider a cycle χ in G . Assume that, for any $\varepsilon > 0$, cycle χ can be realized by a traveller path of travel time $< \varepsilon$. If χ contains a case (iii) edge $e = ((c, d), (d, u))$, then the traveller is able to follow the intersection point $s_t(d, u)$ by using carrier d . Hence, there is no need to use e more than once. We can adapt our algorithm to this case by two minor modifications. The purpose of the following two modifications is to keep the propagation graph free of cycles.

- (i) At that moment when the (first) propagation phase from d to u starts, we finalize all reachable intervals and continue all propagations with new target intervals.
- (ii) While this propagation phase from d to u is active, we do not start any other propagation phase from d to u .

We obtain the following Lemma.

Lemma 4.38. *Suppose, an instance of Traveller's Problem is κ -safe except for a set of cycles Σ such that each cycle $\sigma \in \Sigma$ contains at least one case (iii) edge. Then, Traveller's Problem is decidable.*

Definition 4.39. *Cycle χ in G is called tiny, if*

- χ contains only case (ii) edges, and
- for any $\varepsilon > 0$, there exists a start location and a start time of a traveller path which realizes cycle χ in travel time $< \varepsilon$.

Lemma 4.40. *Tiny cycles can be realized either at most once or infinitely often.*

Proof. Let χ denote a tiny cycle. For each edge e of χ , the time when e is feasible is a (possibly unbounded) interval. Since all edge weights are linear functions, also the set of time points when the traveller can start to run the complete cycle χ is an interval. Let I' denote this *feasibility interval* of χ . Within I' , the travel time needed to realize χ is a linear function h . By the assumption that χ is a tiny cycle, we have $h(t_Z) = 0$ for a point t_Z belonging to the boundary of I' . Let us assume $t_Z = \sup I'$, the other case is symmetric. If the traveller starts to use χ at time $t_Z - t \in I'$, the travel time for the complete cycle χ is given by $h(t_Z - t) = \alpha t$ for some constant $\alpha > 0$. If $\alpha \geq 1$, then χ can be realized at most once. Otherwise, the traveller can perform a trip where he starts to use χ at times $t_Z - t$, $t_Z - (1 - \alpha)t$, $t_Z - (1 - \alpha)^2 t$, ...—this means he can use χ infinitely often. \square

Given any cycle χ , it is easy to determine whether χ can be realized infinitely often. We compute the travel time h of χ as a concatenation of linear functions. Then, we check whether the boundary of the feasibility interval of χ contains a root of h and whether the absolute value of the slope of h is < 1 .

Tiny cycles which can be realized at most once can be handled as cycles containing a case (iii) edge.

Definition 4.41. *A Zeno cycle is a tiny cycle which can be realized infinitely often.*

If χ denotes a Zeno cycle with travel time h , then the time point t_Z which belongs to the boundary of the feasibility interval of χ and fulfills $h(t_Z) = 0$ is called the *Zeno time point* of χ . At time t_Z , all carriers belonging to χ intersect in one common point which is called the *Zeno location* of χ .

Each Zeno time point t_Z requires an edge $e \in E(G)$ to have weight 0 at time t_Z . Since G has $O(n^3)$ edges, the number of Zeno time points is in $O(n^3)$.

Let us reconsider the Zeno cycle shown in Figure 4.2 on page 49. If we take an arbitrary point in the initial triangle as start location instead of s , then it is still possible to reach the Zeno location by using the cycle infinitely often. This behavior is interesting, if we assume the time to run backwards. If the traveller starts at the Zeno location and time runs backwards, then all points of the increasing triangle are reachable.

Figure 4.14 shows how the set of reachable intervals develops before a Zeno time point. At the Zeno time point, the set contains infinitely many reachable intervals. The Zeno location is an accumulation point of this set. Figure 4.15 shows how the set of reachable intervals develops after a Zeno time point, if the Zeno location is reachable at the Zeno time point. Reachable intervals grow which range from the pairwise intersection points of the carriers to those points on the carriers which matched the Zeno location at the Zeno time point.

It is easy to adapt our algorithm to a Zeno cycle χ which is feasible after his Zeno time point t_Z . If the Zeno location is not reachable at time t_Z , no special treatment is

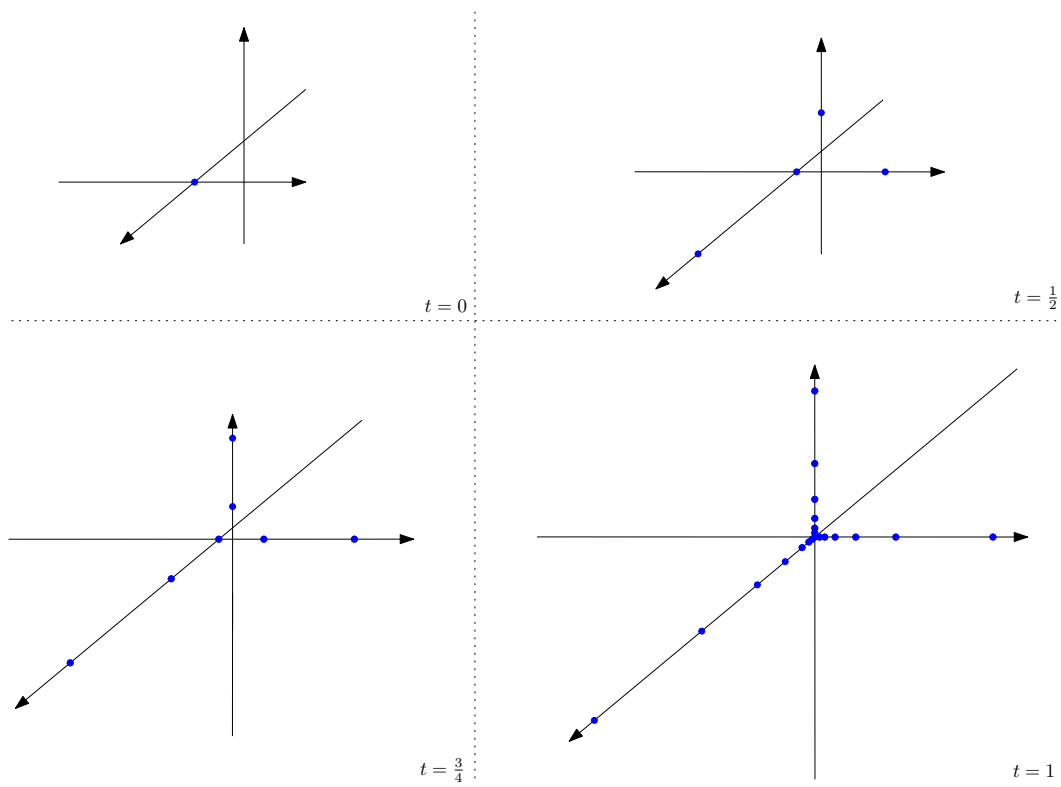
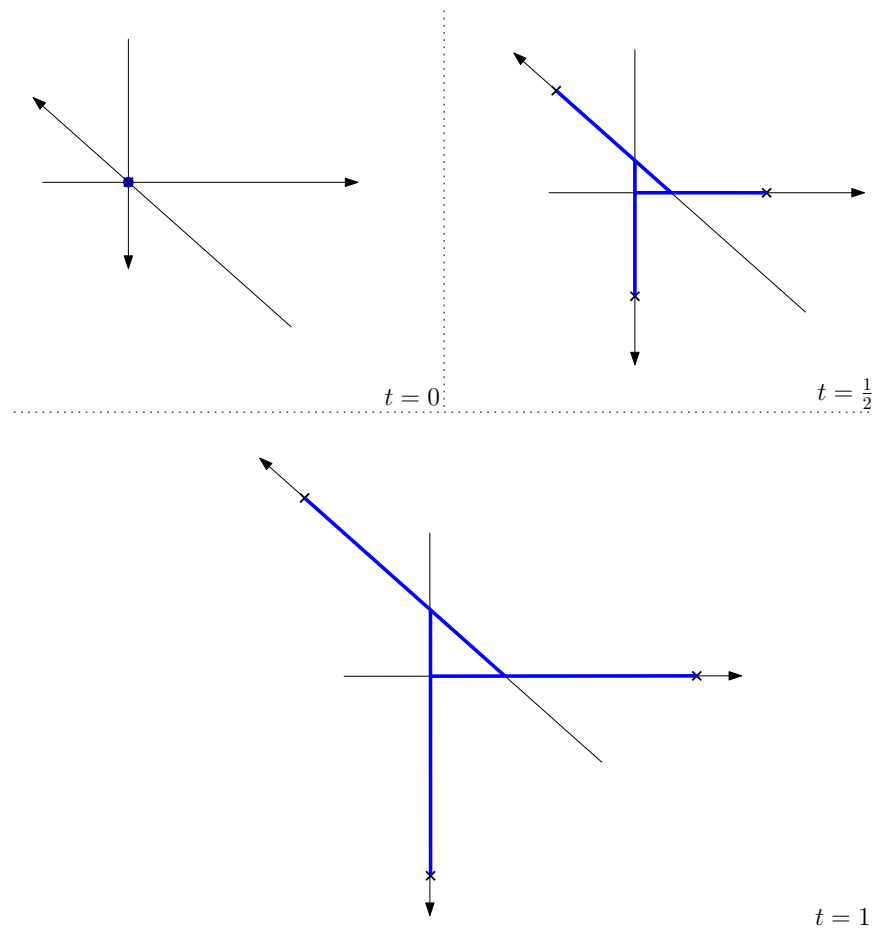


Figure 4.14: Situation before Zeno time point 1. Reachable intervals are blue.



× Carrier points which matched Zeno location at time 0.

Figure 4.15: Situation after Zeno time point 0.

needed. Otherwise, we let reachable intervals as in Figure 4.15 grow. These intervals grow as long as all edges of χ are feasible.

How can we adapt our algorithm, if χ is a Zeno cycle which is feasible before his Zeno time point t_Z ? Let l_Z denote the corresponding Zeno location. To avoid dealing with infinitely many points, we interrupt the propagation shortly before time t_Z . This interruption is restricted to a small environment around l_Z . We choose $\delta > 0$ small enough such that all edges of χ are feasible during the time interval $[t_Z - \delta, t_Z)$. If we interrupt all propagations in an ε -environment of l_Z during the time interval $[t_Z - \delta, t_Z)$, then, at time t_Z , all reachable intervals whose reachability information is missing have distance at most $\varepsilon + \delta v_{\max}$ from l_Z . Here, v_{\max} denotes the maximum velocity of the carriers. Choosing ε and δ small enough ensures that, at time t_Z , the $(\varepsilon + \delta v_{\max})$ -environment around l_Z does not contain any point from any carrier not containing l_Z at time t_Z .

The property that all intervals with missing reachability information are contained in a small environment around l_Z at time t_Z enables us to create *artificial intervals* containing all such intervals. These artificial intervals are created at time t_Z . An example of an artificial interval is shown in Figure 4.16. There is one artificial interval for each carrier containing l_Z at time t_Z . Let R denote any reachable interval containing a location corresponding to a vertex of χ at time $t_Z - \delta$. Then, R is connected to all artificial intervals in the propagation graph. Furthermore, R is also connected to the Zeno location at the Zeno time point which is known to be reachable.

After time t_Z , artificial intervals are propagated where carriers intersect, just like regular intervals; their offspring is marked artificial, too.

At the time bound Ω , we stop the propagation process and inspect the goal point g . If it lies in a “real” interval, we report g to be reachable. If g is not contained in any interval, we output that the goal cannot be reached. If g is contained only in artificial intervals, more work is needed.

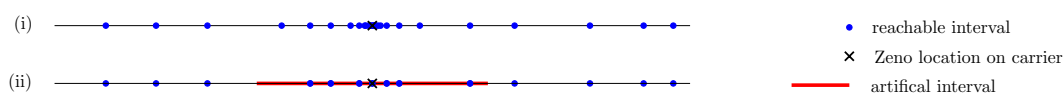


Figure 4.16: (i) Infinitely many reachable intervals, (ii) the algorithm stores only finitely many and artificial intervals containing the missing ones.

For each of the finitely many artificial intervals containing the goal location at time Ω , the propagation graph tells us the corresponding sequence of carrier changes except for the behavior shortly before the Zeno time points. We have to check whether any of these artificial intervals has a path in the propagation graph which can be realized by an s -to- g traveller path.

Let us fix some path ζ in the propagation graph which ends at a propagation phase whose target interval is an artificial interval containing the goal location.

Now, we do a backwards propagation. We let the traveller start at the goal location at time Ω and let time run backwards. We only propagate the reachability as it is possible due to ζ . This backwards propagation is done until we reach the first birth of an artificial interval A_I at Zeno time point t_Z . Let l_Z denote the corresponding Zeno location.

After the backwards propagation we know a closed interval $X \subset A_I$ such that exactly the points belonging to X enable the traveller, if located there at time t_Z , to reach the goal location with a sequence of carrier changes corresponding to ζ .

Figure 4.17 illustrates all known information about the situation at time t_Z for a small example.

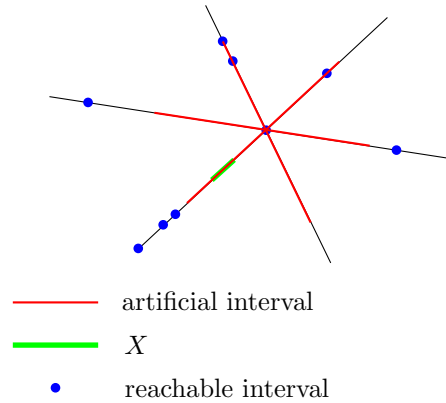


Figure 4.17: Situation at Zeno time point.

We have to find out whether any point of X can be reached at time t_Z . If the Zeno location l_Z belongs to X , then, if ζ can be realized at all, it can be done without using any reachable interval whose reachability information is lost at time t_Z . Hence, we may assume that l_Z does not belong to X .

Now, we continue the forwards propagation from that moment on where we interrupted it in the first run. While continuing the forwards propagation, additional reachable (or artificial) intervals show up. We continue until a time $t_Z - \delta$ where δ is chosen so small that the artificial intervals we have to create around l_Z if we interrupt the propagation at time $t_Z - \delta$ are so small that they do not intersect X . Since the forwards propagation is stopped before the Zeno time t_Z , the arguments leading to Lemma 4.36 ensure that only a finite number of additional reachable (or artificial) intervals shows up.

Spoken in terms of Figure 4.17, continuing the forwards propagation leads to additional blue points and shrinking red intervals. We continue the forwards propagation

until the red intervals are so small that they do not intersect the green one anymore. During the continuing of the forwards propagation, only a finite number of additional blue points shows up.

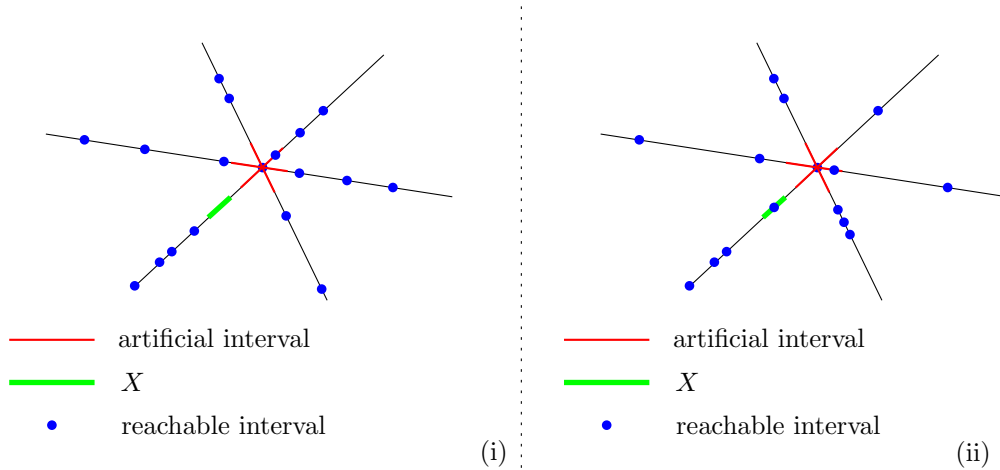


Figure 4.18: After continuing the forwards propagation.

If afterwards no interval (neither reachable nor artificial) intersects X as shown in Figure 4.18 (i), then no traveller path from the start location to the goal location corresponding to ζ exists. If a reachable interval (not an artificial one) intersects X as shown in Figure 4.18 (ii), we can answer the decision problem in the affirmative.

However, if all intervals intersecting X are artificial ones, more work has to be done. In this case, we use every connected component of the intersection of these intervals and X as goal location and repeat the above procedure for all these goals. The number of these new goals is finite, but it seems challenging to find an upper bound for this number which is relevant for the running time of our decision algorithm. Anyway, we do not care about this running time because our only aim is to show decidability.

Each repetition “removes” one Zeno time point. Since the number of Zeno time points is in $O(n^3)$, this algorithm terminates. Afterwards, we know whether ζ can be realized by an s -to- g traveller path. Testing this for all candidate paths ζ in the propagation graph solves the decision problem.

We conclude:

Theorem 4.42. *Traveller’s Problem for restricted instances is decidable.*

4.4.3 Pseudo-polynomial approximation

In this section we provide a pseudo-polynomial approximation algorithm for Traveller’s Problem with bounded carriers in the Euclidean plane. We assume that trav-

eller's innate speed is $v = 1$. Furthermore, we assume that an s -to- g traveller path exists, let P^* denote a path of minimum travel time, W .

In this case, the algorithm we will present computes an s -to- g path of almost minimum travel time. However, this path may only be feasible in a *relaxed model* where the traveller can walk at innate speed $1 + \varepsilon$, and use a carrier although it is a distance of μ away. Both relaxation parameters, ε and μ , can be chosen arbitrarily small.

The algorithm works by discretising time and space.

The time difference between two consecutive discrete time points is

$$\Delta := \frac{\mu}{4 + 8v_{\max}}, \quad (4.3)$$

where v_{\max} denotes the maximum velocity of the carriers.

Space is discretised by a regular grid of width

$$\kappa := \frac{1}{4} \min(\mu, \varepsilon\Delta). \quad (4.4)$$

We align the grid such that the traveller's start location s matches a grid point. At any discrete time point $j\Delta$, the goal location is taken as an extra grid point.

Lemma 4.43. *There is a feasible s -to- g path in the relaxed model which has travel time at most $W + \Delta$, visits grid points at all discrete time points $j\Delta$, and at these time points has distance at most $\frac{\sqrt{2}}{2}\kappa$ from its current carrier.*

Proof. We define W' as the smallest integer multiple of Δ which is at least W , and assume that the optimum path P^* is extended until time W' by letting the traveller follow the goal location during $[W, W']$.

We introduce a function \widehat{P} which maps the discrete time points to the grid points. For any discrete time point $j\Delta$, we define $\widehat{P}(j\Delta)$ as an arbitrary grid point nearest possible to $P^*(j\Delta)$.

By definition of \widehat{P} , we have $\|\widehat{P}(j\Delta) - P^*(j\Delta)\| \leq \frac{\sqrt{2}}{2}\kappa$. The function $\widehat{P} - P^*$ is defined for $t \in \{0, \Delta, 2\Delta, \dots, W'\}$. Let $Q : [0, W'] \rightarrow \mathbb{R}^2$ denote the linear interpolation of $\widehat{P} - P^*$. Since Q interpolates points of the $\frac{\sqrt{2}}{2}\kappa$ -environment around the origin 0, we have $Q(t) \leq \frac{\sqrt{2}}{2}\kappa$ for all $t \in [0, W']$.

Now, we consider the traveller path $P^* + Q$ which uses, at any time, the same carrier as P^* . This path has travel time at most $W + \Delta$ and, at any time, distance at most $\frac{\sqrt{2}}{2}\kappa < \mu$ from its current carrier. The innate speed the traveller needs to follow $P^* + Q$ is at most $1 + \frac{\sqrt{2}\kappa}{\Delta} < 1 + \varepsilon$. Furthermore, at any discrete time $j\Delta$, the point $(P^* + Q)(j\Delta) = \widehat{P}(j\Delta)$ is a grid point. \square

Lemma 4.43 allows us to restrict the path search to paths visiting grid vertices at the discrete time points. Assume, the traveller is located on some grid point p at time $j\Delta$. From there on, which grid points can be reached at time $(j + 1)\Delta$?

The crucial idea is to operate with a set of θ -usable carriers which does not change during the time interval $I = [j\Delta, (j+1)\Delta]$. We declare all carriers having distance at most $\theta := \frac{\sqrt{2}}{2}\kappa + (1+2v_{\max})\Delta$ from p at time $j\Delta$ as θ -usable. During the complete time interval I , we allow the traveller to use all of these carriers and to change arbitrarily between them.

If a carrier has distance greater than $\theta > \frac{\sqrt{2}}{2}\kappa$ from p at time $j\Delta$, this carrier is not needed for the desired path by Lemma 4.43. On the other hand, if the distance from a carrier to p is at most θ at time $j\Delta$, then the distance between p and this carrier cannot exceed $\theta + (1+2v_{\max})\Delta = \frac{\sqrt{2}}{2}\kappa + (2+4v_{\max})\Delta \stackrel{(4.3,4.4)}{<} \frac{3}{4}\mu$ during the complete time interval I .

Hence, no matter how the traveller changes between the θ -usable carriers and walks on them, all extended carrier restrictions are respected. This means we do not have to care about the borders of the θ -usable carriers, and can even imagine them to be unbounded and covering the complete plane.

Lemma 4.44. *Assume, the traveller is located on grid point p at time $j\Delta$. Then, for any grid point q , the following two statements are equivalent:*

- (i) *The point q can be reached at time $(j+1)\Delta$ by using θ -usable carriers in the relaxed model.*
- (ii) *The intersection of the disc of radius $1+\varepsilon$ centered at $\frac{q-p}{\Delta}$ and the convex hull of the velocity vectors of the θ -usable carriers is non-empty.*

Proof. Let w_1, \dots, w_r denote the velocity vectors of the θ -usable carriers. Grid point q can be reached at time $(j+1)\Delta$, if and only if there exist real numbers t_1, \dots, t_r with $0 \leq t_1, \dots, t_r \leq \Delta$ and $t_1 + \dots + t_r = \Delta$ and there exists a vector s corresponding to the traveller's self-movement with $\|s\| \leq (1+\varepsilon)\Delta$ such that:

$$p + t_1 w_1 + \dots + t_r w_r + s = q \tag{4.5}$$

Dividing Equation (4.5) by Δ leads to the equivalent equation

$$\lambda_1 w_1 + \dots + \lambda_r w_r = \frac{q-p}{\Delta} - s' \tag{4.6}$$

where the $\lambda_i = \frac{t_i}{\Delta}$ sum up to 1 and $\|s'\|$ is at most $1+\varepsilon$. Exactly the points belonging to the convex hull $ch(w_1, \dots, w_r)$ can be obtained by the convex combination on the left-hand side of the above equation. Hence, a solution for Equation (4.6) exists if and only if the disc with radius $1+\varepsilon$ centered at $\frac{q-p}{\Delta}$ intersects the convex hull $ch(w_1, \dots, w_r)$. \square

This criterion enables us to check for all grid points whether they are reachable at time $(j+1)\Delta$. By Lemma 4.43, we can restrict the test to grid points having distance at most $\frac{\sqrt{2}}{2}\kappa$ from some carrier at time $(j+1)\Delta$.

Now, the algorithm is straightforward. Based on the information of reachable grid points at time $j\Delta$, grid points reachable at time $(j+1)\Delta$, and then having distance at most $\frac{\sqrt{2}}{2}\kappa$ from their nearest carrier, are computed. For each grid point reachable at time $(j+1)\Delta$, we store a pointer to its predecessor grid point reachable at time $j\Delta$. In this way traveller paths can be reconstructed.

The remaining issue is to analyze the running time of the algorithm.

At each discrete time point the number of reachable grid points is in $O(n \left(\frac{r_{\max} + \mu}{\kappa}\right)^2)$ where r_{\max} denotes the maximum carrier diameter.

The first thing to compute the reachable grid points at time $(j+1)\Delta$ is that we mark all grid points then having distance at most $\frac{\sqrt{2}}{2}\kappa$ from their nearest carrier. This step can be done in time $O(n \left(\frac{r_{\max} + \mu}{\kappa}\right)^2)$, if we mark these points row by row for each carrier. Afterwards, we can check whether any grid point has distance at most $\frac{\sqrt{2}}{2}\kappa$ from some carrier at time $(j+1)\Delta$ in constant running time.

For each grid point p reachable at time $j\Delta$ we have to do the following things. First of all, we determine the set of θ -usable carriers in time $O(n \log L)$, where L denotes the total number of edges of the n carriers. Afterwards, we build the convex hull of the velocity vectors of the θ -usable carriers in $O(n \log n)$. Then, we check for at most $O\left(\left(\frac{v_{\max}\Delta}{\kappa}\right)^2\right)$ grid points q whether they can be reached at time $(j+1)\Delta$ by θ -usable carriers, starting at p at time $j\Delta$, and whether q is marked to have distance at most $\frac{\sqrt{2}}{2}\kappa$ from some carrier at time $(j+1)\Delta$. For each q , this test requires running time $O(\log L)$.

We conclude that, for each discrete time step the total effort is in

$$O\left(n \left(\frac{r_{\max} + \mu}{\kappa}\right)^2 \left(n \log L + \left(\frac{v_{\max}\Delta}{\kappa}\right)^2 \log L\right)\right)$$

The number of discrete time steps is at most $O\left(\frac{W}{\Delta}\right)$.

Plugging in $\Delta \in O\left(\frac{\mu}{v_{\max}}\right)$ and $\frac{1}{\kappa} \in O\left(\frac{\varepsilon + v_{\max}}{\varepsilon\mu}\right)$ leads to the following Theorem.

Theorem 4.45. *A feasible path in the relaxed model with travel time at most $W + \Delta$ can be computed in running time:*

$$O\left(\frac{n^2 \log(L)(r_{\max})^2(v_{\max})^5}{\mu^3 \varepsilon^4} W\right)$$

4.5 Uncountable many paths and the Cantor set

Let us consider the instance shown in Figure 4.19.

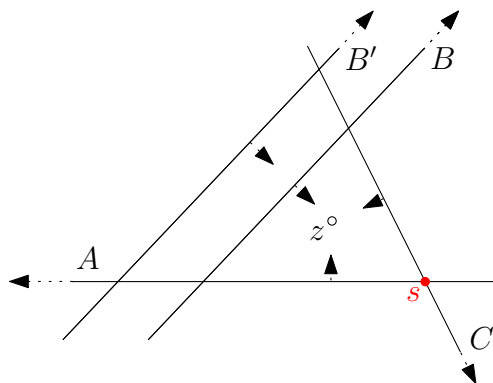


Figure 4.19: Uncountable many traveller paths.

The four line segments A, B', B, C are very long, but bounded. Their velocity vectors have large axial and small lateral components. This causes the segments to intersect in point z at some time t_z . The traveller can reach point z by running the cycle A, B, C infinitely often. Another possibility to get to point z is to run the cycle A, B', C infinitely often. If the traveller runs an infinite sequence of cycles where he changes arbitrarily between A, B, C and A, B', C , it also brings him to z .

Identifying the cycle A, B, C with 0 and the cycle A, B', C with 1, we have a one-to-one correspondence between the infinite binary sequences and the traveller paths bringing the traveller to z . Since the number of infinite binary sequences is uncountable, the same is true for the number of traveller paths to z .

The uncountable number of traveller paths does not imply that the number of reachable points is also uncountable. The cycle travel times for both cycles are linear functions of the time. Let us assume that, if the traveller starts to run the cycle A, B, C at the intersection of A and C at time $t_z - h$, then the arrival time at the intersection of A and C is given by $t_z - \alpha h$. Accordingly, the use of the cycle A, B', C ends at time $t_z - \beta h$, if started at time $t_z - h$.

In this situation, let us consider traveller paths which start at s at time $t - h$. If such a path uses the A, B, C -cycle i times in total and the A, B', C -cycle j times in total, then the path ends at the intersection of A and C at time $t_z - \alpha^i \beta^j h$. The order in which the cycles are used is not important, only the number of usages of A, B, C and A, B', C counts.

Thanks to this property, the uncountable many traveller paths only lead to a countable number of reachable intervals in the above example.

We will establish that the situation is different in three-dimensional space. If the speed into their axial direction is the same for all carriers A, B', B, C , then, for any fixed start time, the travel time of the cycle A, B', C is greater than the travel time of the cycle A, B, C . With α and β defined as above, this means $\beta < \alpha$.

However, if the speed into the axial direction of B' is chosen high enough, then the cycle A, B', C becomes the faster one. We want both cycles to have the same travel time which means we adapt the axial component of B' such that $\alpha = \beta$. By taking into account the axial and lateral components of all carriers, we can even obtain that $\alpha = \beta = \frac{1}{3}$.

Now, we convert the instance of Figure 4.19 to a 3-dimensional one. We replace all carriers A, B, B', C by vertical planes. Figure 4.19 shows the situation from above. We introduce another carrier Z which is a static vertical line through the point z . The only carrier which enables the traveller to change his height is carrier B' . If the traveller starts to use the cycle A, B', C at time $t_Z - h$, then carrier B' increases his height by $\frac{2}{3}h$.

Let us once again identify the cycle A, B, C with 0 and the cycle A, B', C with 1. Then we have a one-to-one correspondence between the infinite binary sequences and the traveller paths to carrier Z .

If the traveller starts his trip at time $t_Z - 1$ at height 0 and uses the binary sequence b_1, b_2, b_3, \dots to reach Z , then he arrives at Z at height $\sum_{i=1}^{\infty} (2b_i) \left(\frac{1}{3}\right)^i$. This means that exactly the points belonging to the well-known Cantor set are reachable on Z . Since the Cantor set is uncountable, we conclude:

Lemma 4.46. *There are instances of Traveller's Problem where the set of reachable points has an uncountable number of connected components.*

4.6 Conclusion

We have introduced a new motion planning problem and shown that its complexity ranges from near linear, in simple cases, to undecidable in higher dimensions. For dimension 2, the problem is NP-hard, but decidable at least for restricted instances. Its "true" complexity in dimension 2 and 3 remains open.

List of Figures

2.1	Lions moving in a 4×4 -grid	10
2.2	Fall-down transformation	12
2.3	Boundary vertices of the cleared area	14
2.4	Illustrating the proof of Lemma 2.6	15
2.5	Eight lions clear the $3 \times 3 \times 3$ -grid	16
2.6	Strategy of flying lions	23
2.7	Speed advantage for the lions	24
3.1	Two participants' upwards cones	30
3.2	Counterexample to Axiom 2	32
3.3	Vertical segments of maximum length	32
3.4	A basis of cardinality 4	35
3.5	Meeting duration determined by 2 participants	36
3.6	Generalization to different speeds	38
3.7	A basis of cardinality n	43
3.8	Number of connected components	43
4.1	Barges on a river	47
4.2	Infinitely many carrier changes	49
4.3	Infinitely many accumulation points	50
4.4	Attic carriers	52
4.5	Equation 4.2 has no solution	59
4.6	No undesired intersections with carrier C_{19}	59
4.7	Intersection graph of C_0, \dots, C_{19}	60
4.8	The affine mapping as a traveller trip	62

4.9	No undesired intersections with C_{19k}	64
4.10	Proving Traveller's Problem NP-hard	66
4.11	Time ω is not enough	70
4.12	Development of reachable intervals	72
4.13	Circular self-dependencies	73
4.14	Situation before Zeno time point	77
4.15	Situation after Zeno time point	78
4.16	Artificial intervals	79
4.17	Situation at Zeno time point	80
4.18	After continuing the forwards propagation	81
4.19	Uncountable many traveller paths	85

Bibliography

- [1] O. Aichholzer, F. Aurenhammer, and B. Palop. Quickest paths, straight skeletons, and the city Voronoi diagram. *Discrete & Computational Geometry*, 31(1):17–35, 2004.
- [2] L. Alonso, A. S. Goldstein, and E. M. Reingold. “Lion and man”: upper and lower bounds. *ORSA Journal on Computing*, 4(4):447–452, 1992.
- [3] Y. Altshuler, V. Yanovski, I. A. Wagner, and A. M. Bruckstein. Efficient cooperative search of smart targets using UAV swarms. *Robotica*, 26(4):551–557, 2008.
- [4] R. Alur, C. Courcoubetis, T. A. Henzinger, and P.-H. Ho. Hybrid automata: an algorithmic approach to the specification and verification of hybrid systems. In R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, editors, *Hybrid Systems I*, volume 736 of *Lecture Notes in Computer Science*, pages 209–229. Springer, 1993.
- [5] J. L. Ambite, G. Barish, C. A. Knoblock, M. Muslea, J. Oh, and S. Minton. Getting from here to there: interactive planning and agent execution for optimizing travel. In *Eighteenth National Conference on Artificial Intelligence*, pages 862–869, 2002.
- [6] E. M. Arkin, J. S. B. Mitchell, and V. Polishchuk. Maximum thick paths in static and dynamic environments. In M. Teillaud, editor, *Symposium on Computational Geometry*, pages 20–27. ACM, 2008.
- [7] B. Aronov and S. Har-Peled. On approximating the depth and related problems. *SIAM Journal on Computing*, 38(3):899–921, 2008.
- [8] E. Asarin, V. Mysore, A. Pnueli, and G. Schneider. Low dimensional hybrid systems: decidable, undecidable, don’t know. Submitted to *Information and Computation*, 2009.
- [9] S. W. Bae, J.-H. Kim, and K.-Y. Chwa. Optimal construction of the city Voronoi diagram. *International Journal of Computational Geometry and Applications*, 19(2):95–117, 2009.

-
- [10] P. Bell and I. Potapov. On undecidability bounds for matrix decision problems. *Theoretical Computer Science*, 391(1-2):3–13, 2008.
- [11] A. BenHassine and T. B. Ho. An agent-based approach to solve dynamic meeting scheduling problems with preferences. *Engineering Applications of Artificial Intelligence*, 20(6):857–873, 2007.
- [12] F. Berger, A. Gilbers, A. Grüne, and R. Klein. How many lions are needed to clear a grid? *Algorithms*, 2(3):1069–1086, 2009.
- [13] F. Berger, A. Grüne, and R. Klein. How many lions can one man avoid? Technical Report 006, Department of Computer Science I, University of Bonn, 2007. <http://web.informatik.uni-bonn.de/I/publications/bgk-hmlcm-07.pdf>.
- [14] F. Berger and R. Klein. A traveller’s problem. In J. Vahrenhold, editor, *European Workshop on Computational Geometry*, pages 93–96, 2010.
- [15] F. Berger and R. Klein. A traveller’s problem. In *Symposium on Computational Geometry*, pages 176–182. ACM, 2010.
- [16] F. Berger, R. Klein, D. Nussbaum, J.-R. Sack, and J. Yi. A meeting scheduling problem respecting time and space. In R. Fleischer and J. Xu, editors, *Algorithmic Aspects in Information and Management*, volume 5034 of *Lecture Notes in Computer Science*, pages 50–59. Springer, 2008.
- [17] F. Berger, R. Klein, D. Nussbaum, J.-R. Sack, and J. Yi. A meeting scheduling problem respecting time and space. *GeoInformatica*, 13(4):453–481, 2008.
- [18] D. Bienstock. Graph searching, path-width, tree-width and related problems. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 5:33–49, 1991.
- [19] T. Böhme and F. Regen. Personal communication, 2009.
- [20] B. Bollobás and I. Leader. Compressions and isoperimetric inequalities. *Journal of Combinatorial Theory, Series A*, 56(1):47–62, 1991.
- [21] P. Brass, K. D. Kim, H.-S. Na, and C.-S. Shin. Escaping offline searchers and isoperimetric theorems. *Computational Geometry*, 42(2):119–126, 2009.
- [22] J. F. Canny and J. H. Reif. New lower bound techniques for robot motion planning problems. In *Symposium on Foundations of Computer Science*, pages 49–60, 1987.
- [23] B. Chazelle and J. Matoušek. On linear-time deterministic algorithms for optimization problems in fixed dimensions. In *Symposium on Discrete Algorithms*, pages 281–290, 1993.

-
- [24] Y.-J. Chiang and J. S. B. Mitchell. Two-point euclidean shortest path queries in the plane. In *Symposium on Discrete Algorithms*, pages 215–224, 1999.
- [25] A. Chun, H. Wai, and R. Y. M. Wong. Optimizing agent-based meeting scheduling through preference estimation. *Engineering Applications of Artificial Intelligence*, 16(7-8):727–743, 2003.
- [26] J. Culberson. Sokoban is PSPACE-complete. In E. Lodi, L. Pagli, and N. Santoro, editors, *Fun with Algorithms*, pages 65–76. Carleton Scientific, 1999.
- [27] A. Dumitrescu, I. Suzuki, and P. Żyliński. Offline variants of the “lion and man” problem: some problems and techniques for measuring crowdedness and for safe path planning. *Theoretical Computer Science*, 399(3):220–235, 2008.
- [28] D. Eppstein. Personal communication, 2007.
- [29] P. Fiorini and Z. Shiller. Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research*, 17(7):760–772, 1998.
- [30] R. J. Fowler, M. S. Paterson, and S. L. Tanimoto. Optimal packing and covering in the plane are NP-complete. *Information Processing Letters*, 12(3):133–137, 1981.
- [31] K. Fujimura and H. Samet. Planning a time-minimal motion among moving obstacles. *Algorithmica*, 10(1):41–63, 1993.
- [32] A. Gajentaan and M. H. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Computational Geometry*, 5(3):165–185, 1995.
- [33] J. Galtier. Lower bounds for γ -cuts on multi-dimensional rectangular grids. Technical Report 36, PRiSM, 1996.
- [34] E. Gervais, H. Liu, D. Nussbaum, Y.-S. Roh, J.-R. Sack, and J. Yi. Intelligent map agents – an ubiquitous personalized GIS. *ISPRS Journal of Photogrammetry and Remote Sensing*, 62(5):347–365, 2007.
- [35] R. Görke, C.-S. Shin, and A. Wolff. Constructing the city Voronoi diagram faster. *International Journal of Computational Geometry and Applications*, 18(4):275–294, 2008.
- [36] J. Gudmundsson. Dagstuhl, November 2009.
- [37] L. J. Guibas and J. Hershberger. Optimal shortest path queries in a simple polygon. *Journal of Computer and System Sciences*, 39(2):126–152, 1989.
- [38] J. E. Hopcroft, J. T. Schwartz, and M. Sharir. On the complexity of motion planning for multiple independent objects; PSPACE-hardness of the “Warehouseman’s Problem”. *The International Journal of Robotics Research*, 3(4):76–88, 1984.

-
- [39] M. N. Huda, E. Kamioka, and S. Yamada. An efficient and privacy-aware meeting scheduling scheme using common computational space. *IEICE - Transactions on Information and Systems*, E90-D(3):656–667, 2007.
- [40] N. R. Jennings and A. J. Jackson. Agent-based meeting scheduling: a design and implementation. *Electronics Letters*, 31(5):350–352, 1995.
- [41] K. H. Johansson, J. Lygeros, S. Sastry, and M. Egerstedt. Simulation of Zeno hybrid automata. In *IEEE Conference on Decision and Control*, volume 4, pages 3538–3543, 1999.
- [42] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computation*, pages 85–103. Plenum Press, 1972.
- [43] R. Klein, D. Nussbaum, J.-R. Sack, and J. Yi. How to fit in another meeting. In *International Conference on Collaborative Computing: Networking, Applications and Worksharing*, pages 1–6. IEEE, 2006.
- [44] D. König. Theorie der endlichen und unendlichen Graphen: kombinatorische Topologie der Streckenkomplexe. *Akademische Verlagsgesellschaft*, 1936.
- [45] A. S. LaPaugh. Recontamination does not help to search a graph. *Journal of the ACM*, 40(2):224–245, 1993.
- [46] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, 1991.
- [47] S. M. LaValle. *Planning algorithms*. Cambridge University Press, 2006.
- [48] F. Leber. Diploma thesis, in preparation, Rheinische Friedrich-Wilhelms-Universität Bonn, 2010.
- [49] D. T. Lee and F. P. Preparata. Euclidean shortest paths in the presence of rectilinear barriers. *Networks*, 14(3):393–410, 1984.
- [50] R. J. Lipton and R. E. Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189, 1979.
- [51] J. E. Littlewood. *A Mathematician’s Miscellany*. Methuen And Company Limited, London, 1953.
- [52] A. Maheshwari, D. Nussbaum, J.-R. Sack, and J. Yi. An $O(n^2 \log n)$ time algorithm for computing shortest paths amidst growing discs in the plane. In T. Tokuyama, editor, *ISAAC*, volume 4835 of *Lecture Notes in Computer Science*, pages 668–680. Springer, 2007.
- [53] J. Matoušek. On geometric optimization with few violated constraints. *Discrete & Computational Geometry*, 14(1):365–384, 1995.

-
- [54] N. Megiddo. The weighted Euclidean 1-center problem. *Mathematics of Operations Research*, 8(4):498–504, 1983.
- [55] K. Mehlhorn, S. Meiser, and R. Rasch. Furthest site abstract Voronoi diagrams. *International Journal of Computational Geometry and Applications*, 11(6):583–616, 2001.
- [56] J. S. B. Mitchell. L_1 shortest paths among polygonal obstacles in the plane. *Algorithmica*, 8(1):55–88, 1992.
- [57] J. S. B. Mitchell, D. M. Mount, and C. H. Papadimitriou. The discrete geodesic problem. *SIAM Journal on Computing*, 16(4):647–668, 1987.
- [58] X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. An approach to the description and analysis of hybrid systems. In R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, editors, *Hybrid Systems I*, volume 736 of *Lecture Notes in Computer Science*, pages 149–178. Springer, 1993.
- [59] D. Nieuwenhuisen, A. Kamphuis, and M. H. Overmars. High quality navigation in computer games. *Science of Computer Programming*, 67(1):91–104, 2007.
- [60] R. J. Nowakowski and P. Winkler. Vertex-to-vertex pursuit in a graph. *Discrete Mathematics*, 43(2-3):235–239, 1983.
- [61] T. D. Parsons. Pursuit-evasion in a graph. In Y. Alavi and D. Lick, editors, *Theory and Applications of Graphs*, volume 642 of *Lecture Notes in Mathematics*, pages 426–441. Springer, 1976.
- [62] R. Penninger. Die Graphensuchvariante des Löwenproblems. Diplomarbeit, Rheinische Friedrich-Wilhelms-Universität Bonn, 2008.
- [63] E. L. Post. A variant of a recursively unsolvable problem. *Bulletin of the American Mathematical Society*, 52:264–268, 1946.
- [64] J. Radon. Mengen konvexer Körper, die einen gemeinsamen Punkt enthalten. *Mathematische Annalen*, 83:113–115, 1921.
- [65] D. Rappaport. Computing the furthest site Voronoi diagram for a set of discs. In *Workshop on Algorithms and Data Structures*, volume 382 of *Lecture Notes in Computer Science*, pages 57–66. Springer, 1989.
- [66] R. V. Rasmussen and M. A. Trick. The timetable constrained distance minimization problem. In J. C. Beck and B. M. Smith, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, volume 3990 of *Lecture Notes in Computer Science*, pages 167–181. Springer, 2006.
- [67] J. H. Reif and Z. Sun. On frictional mechanical systems and their computational power. *SIAM Journal on Computing*, 32(6):1449–1474, 2003.

-
- [68] F. Rinaldi and P. Serafini. Scheduling school meetings. In E. K. Burke and H. Rudová, editors, *Practice and Theory of Automated Timetabling VI*, volume 3867 of *Lecture Notes in Computer Science*, pages 280–293. Springer, 2006.
- [69] P. Santos and H. Vaughn. Where shall we meet? Proposing optimal locations for meetings. Workshop on Map Based Interaction in Social Networks, 2007.
- [70] J. T. Schwartz and M. Sharir. On the “piano movers” problem. II. General techniques for computing topological properties of real algebraic manifolds. *Advances in Applied Mathematics*, 4(3):298–351, 1983.
- [71] S. Sen. Developing an automated distributed meeting scheduler. *IEEE Expert*, 12(4):41–45, 1997.
- [72] E. Shakshuki, H.-H. Koo, D. G. Benoit, and D. L. Silver. A distributed multi-agent meeting scheduler. *Journal of Computer and System Sciences*, 74(2):279–296, 2008.
- [73] M. Sharir and E. Welzl. A combinatorial bound for linear programming and related problems. In A. Finkel and M. Jantzen, editors, *Symposium on Theoretical Aspects of Computer Science*, volume 577 of *Lecture Notes in Computer Science*, pages 569–579. Springer, 1992.
- [74] N. J. A. Sloane. The on-line encyclopedia of integer sequences. <http://www.research.att.com/njas/sequences>.
- [75] J. Wang, C. Niu, and R. Shen. Scheduling meetings in distance learning. In M. Xu, Y. Zhan, J. Cao, and Y. Liu, editors, *Advanced Parallel Processing Technologies*, volume 4847 of *Lecture Notes in Computer Science*, pages 580–589. Springer, 2007.
- [76] E. Welzl. Smallest enclosing disks (balls and ellipsoids). In H. Maurer, editor, *New Results and New Trends in Computer Science*, volume 555 of *Lecture Notes in Computer Science*, pages 359–370. Springer, 1991.

Index

- κ -safe, 73
- θ -usable, 83
- 3SUM-hard, 3, 38–40

- accumulation point, 50, 54, 76
- active row, 25
- artificial interval, 79
- attic carrier, 51
- automata
 - hybrid, 46

- basis change, 31
- basis of an LP-type problem, 31
- boundary vertex, 10

- Cantor set, 86
- carrier, 3, 45
- case (ii) edge, 70
- case (iii) edge, 70
- center k -clustering-problem, 41
- cleared vertex, 10
- clearing number, 10
- closed neighborhood, 9
- compression, 11
- contamination, 2, 8
- continuous Dijkstra, 5, 47
- convex hull, 83
- cycle, 68

- decidability, 4, 5, 47, 65, 81
- dimension of an LP-type problem, 31
- disk
 - enclosing, 29

- edge
 - case (ii) or (iii), 70
- edge weight, 69

- elevator carrier, 51
- enclosing disk, 29

- fall-down transformation, 11
- feasibility interval, 76
- feasible edges, 69
- feasible points, 30
- five-affine-mappings-reachability, 55
- flying lion, 22
- frictional mechanical systems, 5
- frogger, 47
- furthest site Voronoi diagram of circles,
 - 30

- geombase, 39
- goal carrier, 70

- Helly’s theorem, 34
- hybrid automata, 46

- initial basis, 31
- innate speed, 3, 45
- interior vertex, 10
- intersection
 - redundant, 60
- isoperimetric inequality, 11, 20

- layer, 16
 - middle, 18
- lion, 2, 7
 - flying, 22
 - separator, 22
- locality axiom, 31
- LP-type problem, 3, 31

- maximization problem, 31
- meeting scheduling, 3, 27

- middle layer, 18
- monotone vertex set, 12
- monotonicity axiom, 31
- motion planning, 1
- neighborhood, 9
- offline problem, 1
- online problem, 1
- participant, 3, 27
- Post's correspondence problem, 55
- propagation graph, 74
- propagation phase, 71
- pursuit-evasion problem, 7
- pushing process, 11
- reachable, 71
- realization, 68
- redundant intersection, 60
- relaxed model, 82
- restricted instance, 67
- row
 - active, 25
- separator lion, 22
- separator vertex, 22
- simplicial order, 20
- smallest enclosing disk, 29
- sokoban, 1, 5
- tiny cycle, 75
- travel time of a cycle, 68
- traveller, 3, 45
- traveller path, 48
- traveller's problem, 48
- undecidability, 4, 5, 47, 65, 81
- unnecessary carrier change, 57
- vertex
 - boundary, 10
 - cleared, 10
 - interior, 10
 - separator, 22
- violation test, 31
- Voronoi diagram, 30
- Zeno, 47
- Zeno cycle, 76
- Zeno location, 76
- Zeno time point, 76