Exploiting geometric properties in combinatorial optimization

Natalya Usotskaya

This book was typeset by the author using LaTeX.

Cover photo by Shamil Gumirov
Cover design by Shamil Gumirov and Mariya Chaplina
Published by Universitaire Pers Maastricht
ISBN
Printed in the Netherlands by Datawyse

Exploiting geometric properties in combinatorial optimization

Benutten van geometrische eigenschappen in combinatorische optimalisatie

Proefschrift

ter verkrijging van de graad van doctor
aan de Universiteit Maastricht,
op gezag van Rector Magnificus,
Prof. mr. G.P.M.F. Mols,
volgens het besluit van het college van Decanen,
in het openbaar te verdedigen
op woensdag 14 december 2011 om 16.00 uur

door

Natalya Usotskaya

Dit onderzoek werd financieel mogelijk gemaakt door Maastricht Research School of Economics of Technology and Organizations (METEOR).

Acknowledgements

This thesis would never appear without the endless support of some people. I am glad to name them here. First of all, many thanks go to my enormously patient supervisor, Alexander Grigoriev. He was always full of enthusiasm and new ideas about any project.

Part of the research was made in close cooperation with André Berger and Bert Marchal. I am very grateful for the knowledge, ideas and precision they brought into the thesis. My special thanks go to Bert for his patience and for always being there for me.

The unconditional support of my parents led me from the secondary school to this doctoral thesis. Without their understanding and help I would never proceeded so far. Thank you, mom and dad.

Special thanks to my beloved roommate Bas, for charts and the guidance through the Dutch culture and humor. His regular jokes and shiny smiles made rainy days much brighter. He was not the only one who lightened my life during these years. Without mentioning their names I thank all the friends and colleagues from the university, who made my life in Maastricht a very enjoyable and precious time.

A lot of thanks go to Karin and Haydeé for their help with all the paper work. These pretty ladies make the atmosphere in the department pleasant and relaxing.

I thank Rudolf Müller, Jean Cardinal and Ralf Peeters for refereeing the thesis and for all the useful comments they provided to make the text more clear and readable.

The research that forms the basis of this thesis was conducted from 2008 to 2011 at the Quantitative Economics department; it was funded by the department and by Maastricht Research School of Economics of Technology and Organizations (METEOR). This support is gratefully acknowledged.

Natalya Usotskaya
Maastricht, October 2011

Contents

Chapter 1

Introduction

This thesis concerns several interesting combinatorial problems arising in computational geometry and algorithmic graph theory. The common motive in our approach to the problems is to consider them from a "geometric" point of view. Namely, we search for some structural geometric properties helping to attack a combinatorial problem from a different unconventional angle.

1.1 Computational geometry and applications

The first part of the thesis deals with a problem from computational geometry. In computational geometry there are a lot of problems asking to construct a trajectory which is optimal with respect to some natural criteria, such as the distance, time, cost, resource consumption and so on. We refer to the papers [2, 71, 88, 95] as good examples of finding the shortest path in a complex terrain; and to [7, 35, 90] as examples of finding the fastest route. We refer to the shortest paths as length-optimal trajectories and to the fastest routes as time-optimal trajectories. Trajectory optimization problems usually assume a constant speed of the moving object, a *mover*, within some domain. Moreover, many fundamental geometric problems assume that a moving object is a single point in two- or three-dimensional space. Under these two assumptions, length-optimal and time-optimal trajectories are the same. Therefore, dealing with a time-optimization problem, we can apply, for instance, the shortest-path algorithms; see [1, 2].

The situation is completely different when we assume that the speed of the mover depends on its position in space. In this case, length-optimal and time-optimal trajectories can significantly deviate from each other. One example of such a problem is finding fastest paths in weighted subdivisions where the plane is divided into a finite number of regions in each of which the speed is constant, see Daescu [35, 36].

In Part I we consider an even more complicated setting where the speed of the mover changes continuously in one of the system coordinates.

Given a mover in three-dimensional Euclidean space, assume that the absolute value of the mover's speed decreases or increases in one of the space coordinates. This is the real-life setting in any helicopter flight. The higher the altitude of the helicopter is, the lower the atmospheric pressure becomes and, consequently, the lower the air density is. In turn, the reduction in air density will reduce the power available, and then, the maximum speed of the helicopter decreases with its altitude. Though the decreasing concave function of the helicopter's maximum velocity in altitude is quite complex and hardly admits a closed analytical form, it is widely accepted to model it by linear or piece-wise linear functions with a small number of breakpoints, see, e.g. [8]. The complex form of the velocity function is the first difficulty of the helicopter problem. The second difficulty comes from the introduction of obstacles in the terrain. This type of problems is widely considered in computational geometry, see [1, 88, 95]. Usually obstacles are given by the continuous curves in three-dimensional space representing some geographic landscape.

So we can formulate the problem as follows: given two points in three-dimensional space and some continuous curve representing the set of obstacles, find a time-optimal trajectory between them such that the trajectory does not hit an interior of any obstacle. It is noticeable that the helicopter problem with obstacles is a generalization of the classical three-dimensional Euclidean shortest-path problem: if the velocity of the helicopter is constant with respect to its altitude, the problems are equivalent. It is well known that the two-dimensional Euclidean shortest-path problem is polynomially solvable, see Mitchell and Papadimitriou [70], while the three-dimensional problem is NP-hard, see Canny and Reif [27], i.e. computationally intractable. The good news is that we can approximate the optimal solution to any given precision and this can be done in polynomial time, see Agarwal et al. [1].

In this thesis we start with some simplifications of the problem as the most general setting of the helicopter problem is NP-hard and, furthermore, no constant factor approximation algorithm is known. First, we consider the linear velocity function without obstacles, which provides us with so-called *motion primitives* which are the basic optimal elements of the trajectory. The standard technique is to lift from motion primitives to solutions of the general problem so that the trajectory which is optimal in general case is a combination of motion primitives; a typical illustration can be found, for example, in robotics, see, e.g. Chitsaz [29, 30]. So we proceed with the generalizations of the problem in two different ways: one way is to consider the finite set of rectilinear obstacles and solve the basic helicopter problem with linear velocity function under the obstacle constraints. A second way is to generalize the type of the velocity function by introducing a fixed number of breakpoints and solve the general helicopter problem without obstacles. This generalization deals with a more complicated non-uniform medium as in reality the helicopter speed drops more quickly with altitude growth.

The helicopter problem is considered in Chapters 3 and 4. Chapter 3 deals with the basic helicopter problem with rectilinear obstacles. In this chapter we derive the motion primitives and solve the problem with rectilinear obstacles by a polynomial time dynamic programming algorithm.

In Chapter 4 we investigate the general helicopter problem with a piece-wise linear concave velocity function. We reduce the problem to the set of fixed degree multivariable polynomial equations. If the number of breakpoints in the piece-wise linear velocity function is a constant, the resulting system has a fixed number of variables and it can be solved in constant time by algebraic elimination theory; some background is given in Chapter 2, see also Cox [33].

1.2 Graph parameters and algorithms

The second and third part of this thesis is devoted to problems on graphs. Graphs are widely used to represent and to model various problems coming from the real world, for example, telecommunication networks, facility location, project scheduling and so on. For an extended review on applications we refer to [51, 53]. Notice, most of the interesting and practically relevant problems on graphs are also NP-hard, i.e. computationally intractable.

There are several ways to deal with the complexity of the problems. One way is to construct an exact combinatorial algorithm or to use Integer Linear Programming algorithm. Such techniques guarantee the absolute quality of obtained solutions; but they are typically suitable only for small instances of the problem; since the running time of such algorithms is exponential in the input size. Another way to tackle hard problems is to “approximate” or to find some “good enough” solution and guarantee the level of closeness to the optimum.

In this thesis we use the following approach: if the input graph has some structural properties then we explicitly use them to design more efficient algorithms. Many practical problems implicitly satisfy some nice geometric properties; a good example of such a practical problem is a *Traveling Salesman Problem*. The general problem is NP-hard, see Karp [61]; it remains NP-hard even for the case when the cities are in the plane with Euclidean distances, see Papadimitriou [73]. Nevertheless, *Euclidian Traveling Salesman Problem* admits polynomial time approximation schemes, see, e.g. Arora [5]. Another well-known problem with nice geometric properties is *(Euclidian) Steiner Tree Problem* which is NP-complete, see Karp [61], so as *Rectilinear*

Steiner Tree Problem, see Garey and Johnson [48]. But the special case of *Rectilinear Steiner Tree Problem*, so-called *Single-Trunk Steiner Tree Problem* is solved in linear time by Chen et al. [28].

Another way to use this technique is to exploit the structural property of "tree-likeness". It is known that a lot of hard problems are easy on trees. Robertson and Seymour [80, 82] introduced the notion of *tree-decomposition* which represents a graph as some kind of a tree. *Treewidth* is a parameter of graphs; generally speaking, the lower this parameter is, the more tree-like the graph is. Apparently, a lot of hard problems which are easy on trees, are also easy on graphs of small treewidth, see, e.g. [31, 32]. There are various algorithms and heuristics for computing the treewidth. One of the well-known algorithms is a linear time algorithm by Bodlaender [14]. However, it is rarely used in practice because of the huge constant in the running time. Recently, Feige et al. obtained a polynomial time $O(\sqrt{\log tw})$-approximation algorithm [43]. The treewidth of planar graphs can be approximated within a factor $\frac{3}{2}$ by Robertson and Seymour [83] and Seymour and Thomas [87].

In Part II we propose some methods to find and to approximate treewidth. First, we consider the parameters of planar graphs, then we move to the treewidth of general graphs. Planar graphs do not have bounded treewidth and the question whether or not computing the treewidth of planar graphs is *NP*-hard is still open. The treewidth of a planar graph is, however, related to some other parameters of planar graphs, namely its branchwidth and the side size of its largest square grid-minor. The nature of these relationships which provides some intuition on the bounds for the approximation of planar treewidth is examined more closely in Chapter 5.

In Chapter 6 we present another way to find the treewidth of a graph. We consider two Integer Linear Programming formulations for the treewidth problem. The first formulation is proposed by Koster and Bodlaender [65]. We improve this formulation by merging it with the flow metric approach by Bornstein and Vempala [24]. The resulting new ILP cuts some feasible symmetric solutions in the linear relaxation of the ILP by Koster and Bodlaender by new inequalities. Furthermore, we introduce a new ILP formulation for the treewidth problem based on the "geometry" of tree-decompositions. This new formulation allows us to apply the local branching technique by Fischetti and Lodi [46], a novel approach for branch-and-cut algorithms.

Finally, in Part III, Chapter 7, we investigate the problem of excluding a set of graphs as subgraph of the input graph by deleting a minimum number of edges from it. The problem is *NP*-hard on general graph instances. We show that the problem is polynomially solvable on graphs of bounded treewidth. Moreover, we show that an optimal solution of the problem can be approximated using treewidth-based technique when the input graph is planar.

1.3 Publications underlying this thesis

The following set of publications is based on the results of this thesis. The work is done in close cooperation with the respective co-authors.

1. A. Berger, Grigoriev A. and Usotskaya N. *On the Time-Optimal 2D-Trajectories in Non-Uniform Mediums.* METEOR Research Memorandum, **RM/11/031**, Maastricht University, School of Business and Economics.

   The content of this paper is based on Chapters 3 and 4.

2. A. Grigoriev, Ensinck H. and Usotskaya N. *Integer Linear Programming Methods for Treewidth.* METEOR Research Memorandum, **RM/11/030**, Maastricht University, School of Business and Economics.

   Chapter 6 is devoted to the ILP formulations for the treewidth.

3. A. Berger, Grigoriev A. and Usotskaya N. *The Time-Optimal Helicopter Trajectory is a Circle Segment*. In Proc. of 26th European Workshop on Computational Geometry (EuroCG 2010) (2010), pp. 89–92.

   In this paper we presented the first result obtained for the helicopter problem. It is based on Chapter 3.

4. A. Grigoriev, Marchal B. and Usotskaya N. *Algorithms for the Minimum Edge Cover of H-Subgraphs of a Graph*. In Proc. of the 36th Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2010), Lecture Notes in Computer Science, **5901**, Springer (2010), pp.352–464.

   Minimum Edge Cover problem is considered in Chapter 7.

5. A. Grigoriev, Marchal B. and Usotskaya N. *On Planar Graphs with Large Treewidth and Small Grid-Minors*. Electronic Notes in Discrete Mathematics, **32** (2009), pp. 35–43.

   The paper is based on the content of Chapter 5.

Chapter 2

Preliminaries

In this chapter we clarify the terminology used throughout the thesis. Some terms that are utilized only occasionally will be introduced in the subsequent chapters, at the point where they are needed. We commence with the main terminology and results of the calculus of variation and algebraic elimination theory which is widely used in Part I of the thesis. We continue with some terminology and notions that regard standard graph theory used in Parts II and III. Then, some terminology will be introduced to deal with planar graphs and graph minors. Formal definitions of treewidth, tree decomposition and some related notions are followed.

2.1 Calculus of variations

This section contains some well-known definitions of the calculus together with the main definitions and results of the calculus of variations. We refer to [62, 63] and [49, 50] for references.

The following Extreme Value Theorem states:

Theorem 2.1.1. *[62] Let $f : D \longrightarrow \boldsymbol{R}$ be a continuous function in the compact domain D, then f must attain its maximum and minimum value, each at least once on this domain. That is, there exist points m and M in D such that: $f(m) \leq f(x) \leq f(M)$ for every $x \in D$.*

Fermat's Theorem gives the necessary condition which has to be satisfied at the extremum point in the interior of the domain:

Theorem 2.1.2. *[45] Let $f : X \longrightarrow \mathbb{R}$ be a function in the open domain X, $a \in X$ is a local extremum of f. If f is differentiable at a then $f'(a) = 0$.*

These points are called stationary points *of the function.*

As a corollary, global extrema of a function f in the closed domain D occur only at boundaries, non-differentiable points and stationary points in the interior of the domain.

A *functional* is traditionally a map from a vector space to the field underlying the vector space which is usually the field of real numbers. Usually the vector space is a space of functions, thus the functional takes a function as its input-argument. This is the reason why it is sometimes considered as a function of a function. Its use originates in the calculus of variations where one searches for a function which minimizes a certain functional. Often the functionals are given as the integrals from some function and the derivatives of this function, e.g.

$$I(f) = \int_{\Omega} H(f(x), f'(x), f''(x), \ldots)\mu(dx).$$

The following Euler-Lagrange Equation provides the necessary condition a function should satisfy to be the minimizer of a given functional:

Theorem 2.1.3. *[49, 50] Let $S(y) = \int_a^b F(t, y(t), y'(t))dt$ be a functional, where $y : [a,b] \subset \mathbb{R} \to X$ is differentiable and $y(a) = y_a, y(b) = y_b$, y' is the continuous derivative of y and F is a real-valued function with continuous first partial derivatives. If the function $y(t)$ is a stationary point of the functional $S(y)$, then it satisfies the equation:*

$$F'_y(t, y(t), y'(t)) - \frac{d}{dt}F'_{y'}(t, y(t), y'(t)) = 0. \tag{2.1}$$

2.2 Computational algebraic geometry

One of the powerful tools to deal with the systems of multivariable polynomial equations is provided by computational algebraic geometry and commutative algebra, see [33] for details. We briefly introduce main definitions and results of the theory about polynomials of n variables $f(x_1, \ldots, x_n)$:

The set of all polynomials at $x_1, \ldots, x_n$ with coefficients in the field k is denoted $k[x_1, \ldots, x_n]$.

A subset $I \subseteq k[x_1, \ldots, x_n]$ is an *ideal* if it satisfies the following three properties:

- $0 \in I$,
- if $f, g \in I$ then $f + g \in I$,
- if $f \in I$ and $h \in k[x_1, \ldots, x_n]$ then $hf \in I$.

If $f_1, \dots, f_s \in k[x_1, \dots, x_n]$ then

$$\langle f_1, \dots, f_s \rangle = \left\{ \sum_{i=1}^{s} h_i f_i \mid h_1, \dots, h_s \in k[x_1, \dots, x_n] \right\}$$

is an ideal of $k[x_1, \dots, x_n]$. We can think of $\langle f_1, \dots, f_s \rangle$ as consisting of all "polynomial consequences" of the equations $f_1 = \dots = f_s = 0$.

An ideal I is *finitely generated* if there exist $f_1, \dots, f_s \in k[x_1, \dots, x_n]$ such that $I = \langle f_1, \dots, f_s, \rangle$. $f_1, \dots, f_s$ are called a *basis* of I.

The lexicographical order *lex* for the given ordering of the variables $x_1 > x_2 > \dots > x_n$ is determined as follows: $x_1^{\alpha_1} \dots x_n^{\alpha_n} >_{lex} x_1^{\beta_1} \dots x_n^{\beta_n}$ if in the product $x_1^{\alpha_1 - \beta_1} \dots x_n^{\alpha_n - \beta_n}$ the leftmost nonzero entry among $\alpha_i - \beta_i$ is positive. The product $cx_1^{\alpha_1} \dots x_n^{\alpha_n}, c \in k$, is called a *leading term* of the polynomial if $x_1^{\alpha_1} \dots x_n^{\alpha_n}$ is bigger than any other product $y_1 \beta_1 \dots y_n^{\beta_n}$ of the polynomial with respect to the lexicographical order.

The set $\{g_1, \dots, g_s\} \subset I$ is called a *Gröbner basis* of I if and only if the leading term of every element of I is divisible into the leading term of at least one g_i. The Gröbner basis of any nonempty ideal $I \subset k[x_1, \dots, x_n]$ can be found in finite number of steps by *Buchberger's algorithm*, see [33] for details.

Given $I = \langle f_1, \dots, f_s \rangle \subset k[x_1, \dots, x_n]$, the *$l$-th elimination ideal I_l*, $l \geq 1$, is the ideal of $k[x_{l+1}, \dots, x_n]$ defined by $I_l = I \cap k[x_{l+1}, \dots, x_n]$. Thus, I_l consists of all consequences of $f_1 = \dots = f_s = 0$ which eliminate the variables $x_1, \dots, x_l$.

Theorem 2.2.1. *[33] Let $I \subset k[x_1, \dots, x_n]$ be an ideal and let G be a Gröbner basis of I with respect to the lexicographical order $x_1 > x_2 > \dots > x_n$. Then, for every $1 \leq l \leq n$, the set $G_l = G \cap k[x_{l+1}, \dots, x_n]$ is a Gröbner basis of the l-th elimination ideal I_l.*

Thus, the Elimination Theorem 2.2.1 guarantees that we can find a solution of a set of polynomial multivariable equations in a finite number of steps when the highest degree of all polynomials is fixed.

2.3 Graph terminology

A *graph* G is a pair $(V(G), E(G))$, where $V(G)$ is a finite set of *vertices* and $E(G)$ is a set of *edges*.

If there is no ambiguity about which graph G we deal with, we will refer to $V(G)$ and $E(G)$ simply as V and E. Following the convention in graph theory, we use $n(G)$ (or simply n) to denote the number of vertices in G and $m(G)$ (or simply m) to denote the number of edges in G, i.e., $n = |V|$ and $m = |E|$. An edge $e \in E$ is a two-element (multi)-set of vertices from V, i.e., $e \in V \times V$. If an edge e joins two vertices, then these vertices are called the *end vertices* of e. If u and v are the end vertices of the edge e, then e is formally denoted by $\{u, v\}$. For the sake of readability, we will refer to the edge $\{u, v\}$ simply as uv in this thesis. Two edges are *disjoint* if they have no end vertex in common. An edge e and an end vertex of e are said to be *incident* to one another. Two vertices that are joined by an edge are called *adjacent* vertices. The *neighborhood* $N(v)$ of a vertex $v \in V$ in G is the set of vertices adjacent to v in G.

The *degree* $d(v)$ of a vertex $v \in V$ is the number of edges incident to v in G. The minimum degree over all vertices in G is denoted by $\delta(G)$, i.e., $\delta(G) = \min_{v \in V} d(v)$. Similarly, $\Delta(G)$ denotes the maximum degree in G, i.e., $\Delta(G) = \max_{v \in V} d(v)$. An edge $e \in E$ is called *directed* if the pair of vertices denoting the edge is ordered and e is said to be *undirected* if it is represented by an unordered vertex pair. A graph G is undirected if all of its edges are undirected. An edge is called a *self-loop* if its two end vertices coincide. When $E(G)$ is a set rather than a multi-set and does not contain self-loops, G is said to be a *simple* graph. Through the thesis we consider only simple and undirected graphs.

A *path* in a graph G is a sequence of vertices from V such that each vertex in the sequence is adjacent to the next vertex in the sequence. For finite paths, the first vertex is called the *start vertex* and the last vertex is called the *end vertex*. All other vertices in a path are called *internal* vertices. A *cycle* is a path for which the start vertex and the end vertex coincide. A path with no repeated vertices is called a *simple* path, and a cycle with no repeated vertices or edges aside from the necessary repetition of the start and end vertex is a *simple* cycle. A graph is *acyclic* if it does not contain any cycle. The *length* of a path is the number of edges in the path, counting multiple edges multiple times. Two paths in a graph are *independent* if they have no internal vertices in common. The *distance* between two vertices in a graph G is the length of the shortest path in G connecting the two vertices.

If $V' \subseteq V$ and $E' \subseteq E \cap (V' \times V')$, then $G' = (V', E')$ is a *subgraph* of $G = (V, E)$, written as $G' \subseteq G$. If G' is a subgraph of G, then G is a *supergraph* of G'. A subgraph G' of G is said to be induced by V' if E' contains all edges from E for which both end vertices are in V'. The subgraph of $G = (V, E)$ that is induced by $V' \subseteq V$ is denoted by $G[V']$. For $v \in V$, we denote by $G \setminus v$ the graph that is obtained from G by deleting v and all edges incident to v, i.e., $G \setminus v = G[V \setminus \{v\}]$. For $e \in E$, by $G \setminus e$ we denote the graph that results from G when we delete edge e.

A *clique* in G is a subset of V for which all vertices are pairwise adjacent. A clique S in G is called a *maximal* clique if no strict superset of S is a clique. A clique S in G is called a *maximum* clique if there is no clique in G that has bigger size. The size of a maximum clique in G is denoted by $\omega(G)$. A graph on n vertices that forms a clique on n vertices is called a *complete* graph and is denoted by K_n. An independent set in G is a subset of V in which the vertices are pairwise non-adjacent. Maximal and maximum independent sets can be defined analogously to maximal and maximum cliques.

A graph is said to be *connected* if there is a path between each pair of distinct vertices of the graph. A *connected component* S of G is a maximal subgraph of G that is connected, i.e., each supergraph of S that is a subgraph of G is disconnected. Most problems on graphs can be cracked by solving them separately on each connected component of the graph. It is for this reason that we only consider input graphs in this thesis that have exactly one connected component, i.e., we assume that our input graphs are connected. Connected graphs that are acyclic are called *trees*.

2.4 Planar graphs

A graph G is called a *planar graph* if it can be drawn in the plane in such a way that its edges intersect only at shared end vertices. A drawing of a planar graph G in the plane is called a *planar embedding* of G. The regions, isomorphic to open discs, that are bounded by the edges in a planar embedding of a planar graph are called *faces*. If we embed a planar graph on the plane, there is always one face that is unbounded. This is called the *outer face*. All other faces are called *inner faces*. The number of faces in a planar embedding is denoted by the letter f. The next theorem (Euler's formula) implies that for a connected planar graph G, the parameter f does not depend on the embedding of G.

Theorem 2.4.1. *[79] Let G be a connected planar graph with n vertices and m edges. Then for every planar embedding of G it holds that*

$$n - m + f = 2.$$

A graph G is called *outerplanar* (or 1-outerplanar) if there is a planar embedding of G in which each vertex is incident to the outer face. For $k > 1$ a planar embedding is *k-outerplanar* if removing the vertices incident to the outer face results in a $(k-1)$-outerplanar embedding. A graph G is *k-outerplanar* if there exists a k-outerplanar embedding of G, see [6].

In [13], it is shown that the outerplanarity of an arbitrary planar graph can be determined in polynomial time. Later, the following theorem was proven:

Theorem 2.4.2. *[60] Given a planar graph G, the outerplanarity index k of G and a k-outerplanar embedding of G can be found in $O(n^2)$ time.*

2.5 Graph minors

We first explain the notion of an edge contraction in a graph. While doing so, we follow the terminology from [39]. Subsequently, we give a formal definition of a graph minor. Minors can be obtained from a graph by a series of vertex deletions, edge deletions and edge contractions in any order. Minors play an important role in the characterization of many families of graphs.

Informally, contraction of an edge $e = uv$ replaces vertices u and v by a new vertex that is adjacent to all neighbors of both u and v. In this thesis, we restrict ourselves to simple graphs and hence we enforce that after contraction, the new vertex is connected to its neighbors via a single edge and not via a multi-edge. This is illustrated in Figure 2.1.

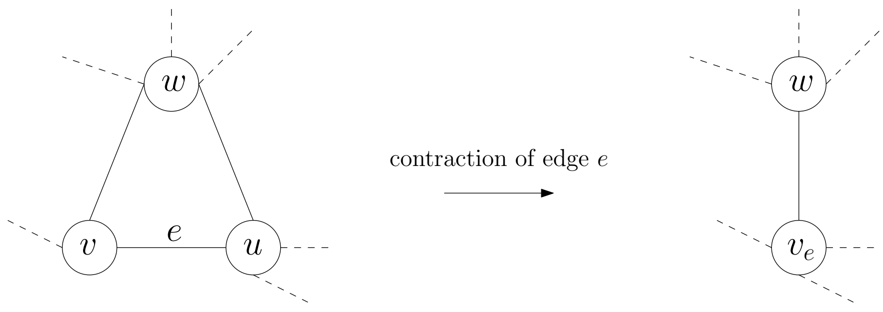


Figure 2.1: Before contraction both u and v are adjacent to w. After contraction of e, the new vertex v_e is connected to w only via a single edge.

As is formalized in the next definition, we use G/e to denote the graph that is obtained from G by contracting edge e.

Definition 2.5.1. *Given graph $G = (V, E)$ and an edge $e = uv \in E$. Then G/e is the graph (V', E') that results from G after contracting edge e, where $V' = (V \backslash \{u, v\}) \cup \{v_e\}$ (with v_e the new vertex) and*

$$E' = \{\ xy \in E \mid \{u, v\} \cap \{x, y\} = \emptyset\ \} \bigcup \{\ v_e x \mid wx \in E \setminus \{e\},\ w \in \{u, v\}\}\,.$$

Two graphs G and H are said to be *isomorphic* if there exists a bijection $f := V(G) \to V(H)$ such that any two vertices u and v are adjacent in G if and only if $f(u)$ and $f(v)$ are adjacent in H.

Definition 2.5.2. *A graph M is a* minor *of G, written as $M \preceq G$, if M is isomorphic to a graph that can be obtained from a subgraph of G applying edge contractions.*

Figure 2.2 shows a graph G, a subgraph H of G and a minor M of G. M can be obtained from H by contracting the dashed edges.

A family $\mathcal{F}$ of graphs is said to be closed under taking graph minors if for every graph G in $\mathcal{F}$, all minors of G are also element of $\mathcal{F}$. Planar graphs and the family of bounded treewidth graphs are examples of graph-minor-closed families.

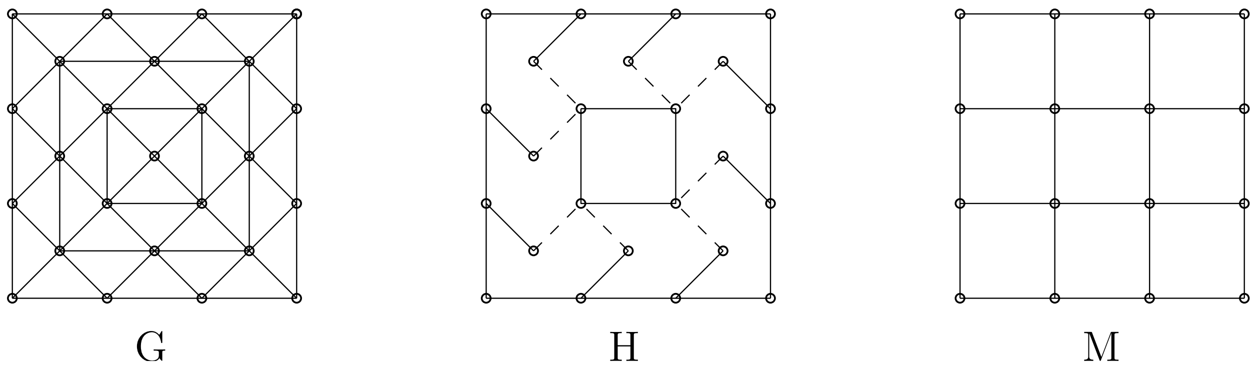


Figure 2.2: graph G, subgraph $H \subseteq G$ and minor $M \preceq G$.

We say that we *subdivide* an edge $e = uv$ in G if we add a new vertex w to G and replace the edge uv by edges uw and vw. A *subdivision* of G can be obtained from G by a series of subdivisions of the edges of G. If a graph M has a subdivision that is isomorphic to a subgraph of G, then M is called a *topological minor* of G. Since in Figure 2.2, graph $H \subseteq G$ is a subdivision of M, M is a topological minor of G.

Kuratowski [66] provided a characterization of planar graphs in terms of two forbidden topological minors; the complete graph K_5 and the complete bipartite graph $K_{3,3}$. Seven years later in [91], Wagner proved that a graph is planar if and only if it does not have K_5 or $K_{3,3}$ as a regular minor. This led him to the conjecture that the set of forbidden minimal minors of any infinite graph-minor-closed family of graphs is finite. A proof of the theorem by Robertson and Seymour was completed in 2004 with the publication of the last in a series of twenty papers (see, among others, [81, 82, 83]) running to over 500 pages and spanning almost 20 years.

2.6 Treewidth

In this section, we formally define treewidth and a few related terms. The notions of treewidth and tree decomposition were introduced by Robertson and Seymour in [81]. Apart from their definitions, we introduce some fundamental lemma's regarding treewidth that are essential for some lemma's and theorems in later chapters of the thesis.

Definition 2.6.1. *[81] A* tree decomposition *of a graph G is a pair (T, X), where T is a tree and $X = (X_t \ : \ t \in V(T))$ is a family of subsets of $V(G)$, with the following properties:*

- $\bigcup_{t \in V(T)} X_t = V(G)$.
- $\forall uv \in E(G),\ \exists t \in V(T)$ *such that* $\{u, v\} \subseteq X_t$.

- *For $t, t', t'' \in V(T)$, if t' is on the unique path in T between t and t'' then $X_t \cap X_{t''} \subseteq X_{t'}$.*

For $t \in V(T)$, the set X_t is also referred to as a bag *of the tree decomposition (T, X).*

The *width* of the tree decomposition (T, X) is

$$\max_{t \in V(t)} (|X_t| - 1).$$

Definition 2.6.2. *[81] The* treewidth *$tw(G)$ of graph G is the minimum width over all tree decompositions of G.*

The following two lemmas are “folklore”.

Lemma 2.6.3. *Let (T, X) be a tree decomposition of graph G. Then for any clique S in G, there is a bag X_t in (T, X) for which $S \subseteq X_t$.*

Lemma 2.6.4. *If $M \preceq G$, then $tw(M) \leq tw(G)$.*

A lot of research has been dedicated to the fixed parameter case for treewidth, i.e., check whether the treewidth of a graph is at most some constant w and if so, return a tree decomposition of width at most w. For an overview of this work we refer to [16]. Finally, the following result was obtained.

Theorem 2.6.5. *[14] Given a graph of treewidth at most w, a tree decomposition of width at most w can be obtained in linear time.*

As we noticed above, from a practical viewpoint the algorithm is only useful for very low values of w.

The following theorem relates the outerplanarity of a graph to its treewidth.

Theorem 2.6.6. *[15] The treewidth of a k-outerplanar graph is at most $3k - 1$.*

An edge which joins two vertices of a cycle but is not itself an edge of the cycle is called a *chord* of that cycle. A *chordless* cycle in G is an induced cycle in G, i.e., a cycle that forms an induced subgraph of G. A graph is called *chordal* (or *triangulated*) if it does not contain chordless cycles of length greater than 3, i.e., if all induced cycles in the graph are 3-cycles (also called triangles). A graph can be transformed into a chordal graph by adding edges to it up to the point where every cycle contains a chord. Edges that are added to achieve chordality are called *fill-in edges*. We denote the set of fill-in edges by F.

Definition 2.6.7. *A graph $H = (V, E \cup F)$ is called a* chordalization *(or* triangulation*) of $G = (V, E)$ if H is chordal.*

The *width* of a chordal graph H is equal to $\omega(H) - 1$, i.e., the size of the largest clique in H minus one. Using the notion of chordalization, treewidth can be alternatively defined in the following way.

Theorem 2.6.8. *see, e.g. [16] The treewidth $tw(G)$ of graph G is the minimum width over all chordalizations of G.*

The following result concerning chordal graphs holds

Lemma 2.6.9. *[84] Given a chordal graph G, the size of a largest clique in G and hence $tw(G)$ can be determined in polynomial time.*

An *ordering* of the vertex set V is a bijection $\alpha : \{1, 2, \ldots, n\} \longleftrightarrow V$. If α is an ordering on n vertices, then we will also refer to α as $\alpha(1)\alpha(2)\ldots\alpha(n)$. A chordalization of $G = (V, E)$ can be obtained using vertex ordering α of V by application of Algorithm 2.1. In words, we run through the vertex ordering and turn the higher

Algorithm 2.1: chordalization

```
Input: graph $G = (V, E)$ and vertex ordering $\alpha$
Output: chordalization of $G$

for $i = 1$ to $n - 2$ do
    make $S_i = \{ v \in V \mid v \in N(\alpha(i))$ && $\alpha(v) > i \}$ a clique by adding fill-in edges
    (if necessary);
```

ordered neighbors of the vertex at hand into a clique by adding fill-in edges to G. Note that the added fill-in edges also define neighbor relations in subsequent steps of the algorithm.

The process of turning the higher ordered neighbors of vertex $\alpha(i)$ into a clique is often called the *elimination* of vertex $\alpha(i)$. In the context of chordalizations, vertex orderings are therefore often referred to as *elimination orderings*. An elimination ordering α on the vertices of G is called *perfect* if during Algorithm 2.1 no fill-in edges are added to G, i.e., if for all vertices $v \in V$ the set of higher ordered neighbors already forms a clique in G.

A vertex v in a graph G is called *simplicial* if $N(v)$ induces a clique in G. It was shown in [67] that every non-empty chordal graph contains at least one simplicial vertex. Another classical fact is that chordal graphs are recursively simplicial, i.e., they contain a simplicial vertex and after removing this simplicial vertex, the subgraph is still simplicial. A perfect elimination ordering α of a chordal graph G can therefore be obtained by repeatedly selecting a simplicial vertex of G as the next vertex in α and deleting it from G.

Part I

Computational Geometry and Applications

Chapter 3

Basic Helicopter Problem with Rectilinear Obstacles

The classical trajectory optimization problems in computational geometry belongs to the following type: given a moving object, a *mover*, together with the velocity function and some obstacles find the time-optimal trajectory from one point to another such that this trajectory does not hit an interior of any obstacle. Furthermore, these problems usually assume that the speed of the mover is constant and that the mover is a single point. Under these two assumptions, length-optimal and time-optimal trajectories are the same. Therefore, dealing with a time-optimization problem, we can apply, for instance, the shortest-paths algorithms; see, e.g., [1, 2].

The situation is totally different when we assume that the speed of the mover depends on the position in the space. In this case, length-optimal and time-optimal trajectories can deviate from each other significantly. One example for such a problem is finding shortest paths in weighted regions [35, 36].

In two following chapters we investigate such fastest path problem where the speed changes continuously in one of the space's coordinates, namely we consider the *helicopter problem*. The maximum velocity of any helicopter drops with altitude growth. Though the concave function of the helicopter's maximum velocity in altitude is quite complex and hardly admits a closed analytical form, it is widely accepted to approximate it by linear or piece-wise linear concave functions with just few breakpoints, see, e.g. [8]. Therefore, we can formulate the *basic helicopter problem* as follows: given is a source point A and a destination point B in three-dimensional Euclidean space. No obstacles (also no ground level) are present. The absolute value of the mover's speed decreases linearly in altitude, i.e., $v(y) = max\{v_0 - qy, 0\}$, where $v_0 > 0$ is some speed intercept (for instance, the helicopter's maximum velocity at a ground level) and $q \geq 0$ is the velocity degradation rate. One has to find a time-optimal trajectory to fly the mover (*helicopter*) from A to B. Notice that this three-dimensional problem can be easily reduced to the two-dimensional problem, as the time-optimal trajectory clearly belongs to the plane orthogonal to the surface and containing points A and B.

The *helicopter problem with polyhedral obstacles* is formulated as follows: given points A and B in three-dimensional Euclidean space, along with a set of polyhedral obstacles, find a time-optimal trajectory to fly the helicopter from A to B without hitting an interior of any obstacle. The problem is NP-hard as a generalization of the three-dimensional Euclidean shortest-path problem [27]: if all the velocity degradation rates are equal to zero, the problems are equivalent. It is well known that the two-dimensional Euclidean shortest-path problem is polynomially solvable [70], the three-dimensional problem is NP-hard [27], but admits polynomial time approximation schemes [1].

Notice that one can derive a polynomial-time approximation scheme to the three-dimensional helicopter problem with obstacles by discretizing/scaling the space and constructing a weighted complete graph, where the edge weight is the time to travel between two vertices using the straight line trajectory. The presence of obstacles can be taken into account setting the edge weight to positive infinity if the straight line between two vertices hits interior of an obstacle. Now we can search for the shortest path in the obtained weighted graph.

In the time-optimization setting, to tackle the problem with obstacles, one might need a complete characterization of the set of optimal solutions to the basic problem without obstacles, the set of, so-called, *motion primitives*. Typical illustrations of lifting from motion primitives to solutions to the general problem can be found, for example, in robotics, see [29, 30]. In this chapter we derive motion primitives for the basic helicopter problem. We show that the basic helicopter problem is reducible to the l'Hôpital's problem [9], one of the most studied problems in geometrical optics and mechanics. We use Euler-Lagrange equations from calculus of variations as a tool to tackle the problem. One can see this also as a variant of the Pontryagin Maximum (Minimum) Principle [76]. This type of techniques is quite common in optimal control theory in general, and in robotics in particular; see, e.g., [10, 11]. We obtain that the trajectory following a certain circle segment with endpoints A and B is the time-optimal trajectory in a class of continuously differentiable functions. We generalize results for the basic helicopter problem in the following way: we show that the basic helicopter problem with rectilinear obstacles is solvable in polynomial time by a dynamic programming algorithm. We assume a computational model with infinite precision real arithmetic to avoid the comparison of lengths of two paths and so on; this approach is quite common, see Mitchell and Sharir [71].

The chapter is organized as following: in Section 3.1 we derive the motion primitives to the basic helicopter problem. In Section 3.2 we introduce the rectilinear obstacles as a continuous curve in two-dimensional space and reduce the problem with obstacles to a dynamic program. In particular, we construct a weighted graph such that the shortest path in this graph corresponds to the time-optimal helicopter trajectory with obstacles. The last section contains the results and open questions.

3.1 Solution of the basic helicopter problem

In this section we consider the basic helicopter problem; no obstacles and no ground level are present. The velocity is given by the linear function $v(y) = max\{v_0 - qy, 0\}$, where $v_0 > 0$ is some speed intercept (normally we assume that it is the helicopter's maximum velocity at the ground level) and $q \geq 0$ is the velocity degradation rate. The problem is to find the time-optimal trajectory from one point to another in three-dimensional space. As mentioned above, this problem is reducible to the problem in two-dimensional space, thus we consider a starting point $A = (x_1, y_1)$ and an end point $B = (x_2, y_2)$ of the trajectory.

First, we notice that if the points $A = (x_1, y_1)$ and $B = (x_2, y_2)$ lay on the same vertical line (i.e., they share the same $x-$coordinate: $x_1 = x_2$), then the time-optimal trajectory is just a piece of the straight line $L_{AB} = \{x \mid x = x_1\}$ between A and B. We omit the proof as it is straightforward. We also assume that $q > 0$; for otherwise we are in the well-studied setting of the classical Euclidean shortest-path problem. Thus, from now on we consider the case where $x_1 \neq x_2$ and $q > 0$. We show that in this case the circle segment is the unique time-optimal trajectory. The first observation is about the convexity of any time-optimal trajectory.

Lemma 3.1.1. *For any two points A and B in $\mathbb{R}^2 \cap \{(x, y) : \ v_o - qy > 0\}$, a time-optimal trajectory between A and B is a convex function of x.*

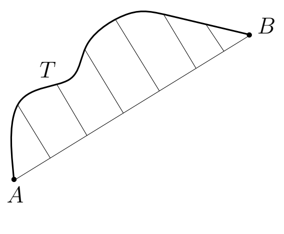


Figure 3.1: Straight line L_{AB} is quicker than any trajectory T above it

Proof. The proof is based on a comparison of the straight line segment between any two points A and B and any trajectory above this line segment. We discretize the line and the path by the lines perpendicular to the straight line segment L_{AB}, see Figure 3.1 for details. The distance to travel along the straight line is smaller and the velocity increases with the drop of altitude. Hence, the straight line trajectory is better than any trajectory above it, so the time-optimal trajectory between A and B must be a convex function of x. $\square$

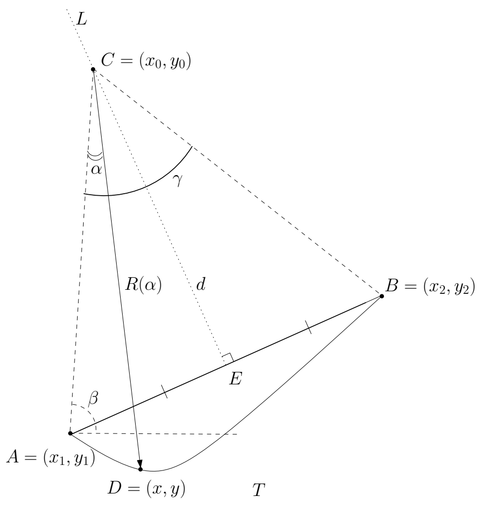


Figure 3.2: Convex trajectory T in the polar system of coordinates

By Lemma 3.1.1, we can restrict our search for time-optimal trajectories to the class of convex functions in x. Let T be an arbitrary continuously differentiable convex trajectory between A and B. Let L be the line equidistant from A and B. Since T is convex in x, it is possible to observe every point of T from any point C on L that lies above the line segment AB. We consider the trajectory T in the polar system of coordinates with observation point $C = (x_0, y_0)$, where $C \in L$ is chosen such that the velocity in C is 0, i.e., $y_0 = \frac{v_0}{q}$. For illustration see Figure 3.2. We refer to the angle between CA and the x-axis as β and to the angle between CA and CB as γ. Consider an arbitrary point $D = (x, y)$ of the trajectory T. In the chosen polar system of coordinates, the point D is completely determined by α and $R(\alpha)$, where α is the angle between CA and CD, and $R(\alpha)$ is the length of the interval CD:

$$x = x_0 - R(\alpha)\cos(\alpha + \beta), \tag{3.1}$$

$$y = y_0 - R(\alpha)\sin(\alpha + \beta). \tag{3.2}$$

First, we write the integral which represents the time needed to travel along T. For a sufficiently small piece of the trajectory, we may assume that the velocity v remains constant within the piece. Let the length of the piece be denoted by Δl, then the time to travel along the piece is $\Delta t \approx \frac{\Delta l}{v}$. The velocity v is completely determined by the altitude of $D = (x, y)$. Therefore, by Equation (3.2), we have $v = v_0 - qy = qR(\alpha)\sin(\alpha+\beta)$, as $v_0 - qy_0 = 0$ by the choice of C. We know that for continuously differentiable trajectories the length of the piece Δl is determined by $\Delta l = \sqrt{R'^2(\alpha) + R^2(\alpha)}\Delta\alpha$, see, e.g. [63]. Therefore, the time needed to travel along T is determined by the following integral:

$$t_T = \int_0^\gamma \frac{\sqrt{R'^2(\alpha)+R^2(\alpha)}\,d\alpha}{qR(\alpha)\sin(\alpha+\beta)} = \int_0^\gamma \sqrt{\left(\frac{R'(\alpha)}{R(\alpha)}\right)^2 + 1}\frac{d\alpha}{q\sin(\alpha+\beta)}. \tag{3.3}$$

Equation (3.3) allows us to proof the following theorem about the unique time-optimal trajectory between A and B.

Theorem 3.1.2. *Let C be the intersection point of the line $y = \frac{v_0}{q}$ and the line equidistant from A and B. The segment T_{AB} of the circle with center C and radius $R = |CA| = |CB|$ is a time-optimal trajectory between A and B. The time needed to travel along T_{AB} is*

$$t_{opt} = t_{T_{AB}} = \frac{1}{q}\int_0^\gamma \frac{d\alpha}{\sin(\alpha+\beta)} = \frac{1}{q}\ln\left|\frac{\tan\frac{\beta}{2} + \tan\frac{\gamma}{2}}{\tan\frac{\beta}{2} - \tan^2\frac{\beta}{2}\tan\frac{\gamma}{2}}\right|, \tag{3.4}$$

where β is the angle between CA and the x-axis, and γ is the angle between CA and CB.

Proof. Equation (3.3) gives the time needed to go alone the arbitrary convex trajectory T. Notice, this integral is well defined as $0 < \alpha + \beta < \pi$ for any $\alpha \in [0, \gamma]$ (β and γ are two of the three angles of the triangle). Furthermore, for the trajectory T we have that $R(0) = R(\gamma) = R = |CA| = |CB|$, as the point C is equidistant from A and B.

Now, we replace $R(\alpha)$ with the function $f(\alpha) = \ln(R(\alpha))$. Since $R(\alpha)$ is continuously differentiable, the function $f(\alpha)$ is continuously differentiable as well. Furthermore, $f'(\alpha) = \frac{R'(\alpha)}{R(\alpha)}$, and the following boundary conditions hold: $f(0) = f(\gamma) = \ln R$, where $R = |CA| = |CB|$. For the function $f(\alpha)$ the travel time is calculated as follows:

$$t_T = \frac{1}{q}\int_0^\gamma \sqrt{1 + f'^2(\alpha)}\frac{d\alpha}{\sin(\alpha+\beta)}. \tag{3.5}$$

Now, we have to find all minimizers of the functional (3.5).

Notice that the trajectory is a circle segment if and only if $R(\alpha)$ is constant, which is possible if and only if the function $f(\alpha) = \ln(R(\alpha))$ is also a constant. Therefore, we can concentrate on Equation (3.5) proving that $f(\alpha) = const$ is the unique minimizer of the functional.

1) We prove first that the circle segment with the center C and radius $R = |CA|$ is a time-optimal trajectory.

Consider an arbitrary trajectory T. $f'^2(\alpha) \geq 0$ for any T, therefore, the following lower bound exists:

$$t_T \geq \frac{1}{q}\int_0^{\gamma} \frac{d\alpha}{\sin(\alpha+\beta)}.$$

On the other hand, the last integral represents the time needed to travel along the circle segment T_{AB} with the center C and the constant radius $R = |CA|$. Therefore, the derived circle segment is at least as good as any other trajectory.

2) Now, we show that T_{AB} is the unique time-optimal trajectory in the class of continuously differentiable functions. Consider the functional (3.5) with the integrand $F(\alpha, f, f') = \frac{\sqrt{1+f'^2(\alpha)}}{\sin(\alpha+\beta)}$. Suppose $f(\alpha)$ is the minimizer of the functional. Therefore, by Theorem 2.1.3, the Euler-Lagrange Equation (2.1) holds for $f(\alpha)$, because it is a necessary condition for the stationary point of any functional.

   We obtain the equation $\frac{d}{d\alpha}(F'_{f'}) = 0$ as $F'_f = 0$. Hence,

$$F'_{f'} = \frac{f'(\alpha)}{\sqrt{1+f'^2(\alpha)}\sin(\alpha+\beta)} = const. \tag{3.6}$$

   Since $\sqrt{1+f'^2(\alpha)} \geq 1$ and $\sin(\alpha+\beta) > 0,\ (0 < \alpha+\beta < \pi)$, the denominator in Equation (3.6) is always positive. So the sign of $f'(\alpha)$ depends only on the sign of the constant. If $const = 0$ then $f'(\alpha) = 0$ and $f(\alpha)$ is a constant function, so as $R(\alpha)$. This minimizer is the circle segment described above. If $const > 0 (< 0)$ then $f(\alpha)$ is strictly increasing (decreasing), but this is the contradiction to the boundary condition $f(0) = f(\gamma) = \ln\ R$.

   Hence, there are no other time-optimal trajectories but the circle segment T_{AB} with the center C and the radius $R = |CA| = |CB|$.

The time needed to travel along the circle segment T_{AB} can be computed as the value of the integral:

$$\begin{aligned} t_{opt} &= \frac{1}{q}\int_0^{\gamma}\frac{d\alpha}{\sin(\alpha+\beta)} = \frac{1}{q}\int_{\beta}^{\beta+\gamma}\frac{d\tau}{\sin\tau} = \frac{1}{q}\int_{\tan\frac{\beta}{2}}^{\tan\frac{\beta+\gamma}{2}}\frac{dt}{t} = \\ &= \frac{1}{q}\ln\left|\frac{\tan\frac{\beta+\gamma}{2}}{\tan\frac{\beta}{2}}\right| = \frac{1}{q}\ln\left|\frac{\tan\frac{\beta}{2}+\tan\frac{\gamma}{2}}{\tan\frac{\beta}{2}-\tan^2\frac{\beta}{2}\tan\frac{\gamma}{2}}\right|. \end{aligned}$$

Here, $\tan\frac{\gamma}{2}$ and $\tan\frac{\beta}{2}$ can be determined in terms of the coordinates $A = (x_1, y_1)$ and $B = (x_2, y_2)$. $C = (x_0, y_0)$ has an altitude $y_0 = \frac{v_0}{q}$. Since C belongs to the line L

equidistant from A and B, we have the following set of values:

$$\begin{aligned}
x_0 &= -\frac{y_2-y_1}{x_2-x_1}y_0 - \frac{x_1^2+y_1^2-x_2^2-y_2^2}{2(x_2-x_1)}, & (3.7)\\
R &= |CA| = |CB| = \sqrt{(x_0-x_1)^2+(y_0-y_1)^2}, & (3.8)\\
d &= |CE| = \sqrt{\left(x_0-\frac{x_1+x_2}{2}\right)^2+\left(y_0-\frac{y_1+y_2}{2}\right)^2}. & (3.9)
\end{aligned}$$

Now we determine the tangents of β and γ in the following way: $\tan\frac{\gamma}{2} = \frac{\sqrt{R^2-d^2}}{d}$. Since $\sin\beta = \frac{y_0-y_1}{R}$ and $\cos\beta = \frac{x_0-x_1}{R}$, then $\tan\frac{\beta}{2} = \frac{\sin\beta}{1+\cos\beta} = \frac{y_0-y_1}{R+x_0-x_1}$. □

To demonstrate the reducibility of the considered problem to the l'Hôpital's problem, we present the time needed to travel along the trajectory T in the following way:

$$t_T = \int_{y_1}^{y_2} \frac{\sqrt{1+\dot{x}^2(y)}}{v_0-qy}dy,$$

as we have to go from the altitude y_1 to the altitude y_2 along the trajectory $x(y)$, when speed depends linearly on altitude, so $v(y) = v_0 - qy$, and the length of the trajectory piece is $\Delta l = \sqrt{1+\dot{x}^2(y)}$, see [9]. This integral can be seen as a solution to the l'Hôpital's problem, see [9], Chapter 3.

3.1.1 Continuity of the time function

In this subsection we show the continuity of the time function t_{opt}. Without loss of generality we consider the points $A = (0, y_1)$ and $B = (x, y_2)$. It is clear that the time function $t_{opt}(x)$ is continuous for all values of x not equal to zero as it is a composition of continuous functions. So we only have to check the continuity at point $x = 0$. We show below that if x goes to zero then the value of $t_{opt}(x)$ goes to the time needed to go along the vertical segment from the altitude $y = y_1$ to the altitude $y = y_2$, thus the time function is continuous.

We rewrite the time function $t_{opt}(x)$ in the following way:

$$t_{opt}(x) = \frac{1}{q}\ln\left|\frac{\tan\frac{\beta(x)}{2}+\tan\frac{\gamma(x)}{2}}{\tan\frac{\beta(x)}{2}-\tan^2\frac{\beta(x)}{2}\tan\frac{\gamma(x)}{2}}\right| = \frac{1}{q}\ln\left[\frac{1+f(x)}{1-f(x)}\right],$$

where $f(x) = \frac{x^2+(y_2-y_1)^2}{\sqrt{((y_2-y_1)(2y_0-y_1-y_2)+x^2)^2+4x^2(y_0-y_2)^2}}$. If x goes to zero then $f(x)$ behaves in the following way

$$f(x) \underset{x\to 0}{\longrightarrow} \frac{(y_2-y_1)^2}{\sqrt{((y_2-y_1)(2y_0-y_1-y_2))^2}} = \frac{y_2-y_1}{2y_0-y_1-y_2}.$$

Thus,

$$\frac{1+f(x)}{1-f(x))} \underset{x\to 0}{\longrightarrow} \frac{2y_0 - y_1 - y_2 + y_2 - y_1}{2y_0 - y_1 - y_2 - y_2 + y_1} = \frac{2y_0 - 2y_1}{2y_0 - 2y_2} = \frac{v_0 - qy_1}{v_0 - qy_2}.$$

We conclude that

$$t_{opt}(x) \underset{x\to 0}{\longrightarrow} \frac{1}{q} \ln \left[\frac{v_0 - qy_1}{v_0 - qy_2}\right].$$

But this is exactly equal to the time needed to go along the vertical linear segment from the altitude $y = y_1$ to the altitude $y = y_2$, because the time can be presented as a following integral:

$$\int_{y_1}^{y_2} \frac{dy}{v_0 - qy} = \frac{1}{q} \ln \left[\frac{v_0 - qy_1}{v_0 - qy_2}\right].$$

Hence, the time function $t_{opt}(x)$ is continuous in the interval $[0, +\infty]$.

3.2 Introduction of rectilinear obstacles

In this section we consider the basic helicopter problem with rectilinear obstacles in 2D. The obstacles are given as a sequence of points $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, so that every pair of consecutive points have exactly one common coordinate (either x_i or y_i) and these same coordinates alternate, see Figure 3.3. We assume that the helicopter goes from the left to the right without any loops; furthermore, the start point $A = (x_1, y_1)$ and the end point $B = (x_2, y_2)$ belong to the set S; otherwise without loss of generality we "draw" an additional obstacle to the left of the terrain with the corner A and add A to S. Here and below we refer to the straight line segment between any two points K and M as L_{KM} and to the time-optimal circle segment between K and M from Theorem 3.1.2 as T_{KM}. The center and the radius of T_{KM} are denoted by $C_{KM} = (x_0, y_0)$ and R_{KM}. The values of x_0 and R_{KM} are determined by Equations (3.7) and (3.8) respectively; $y_0 = \frac{v_0}{q}$.

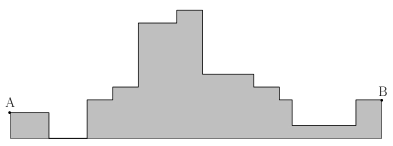


Figure 3.3: Rectilinear obstacles: the shaded area represents the forbidden territory

3.2.1 Dynamic programming algorithm

The idea for the algorithm computing a time-optimal two-dimensional helicopter trajectory in presence of rectilinear obstacles is to find all potential breakpoints such that the trajectory changes the motion primitives in these points. If the amount of potential breakpoints is polynomial then a simple dynamic program can solve the problem in polynomial time. Obviously, the points of the set S are already potential breakpoints. Are there any other points? The answer is "yes" and it is given by the following lemma. It considers the time-optimal trajectory with respect to the "one step" obstacle, see Figure 3.4.

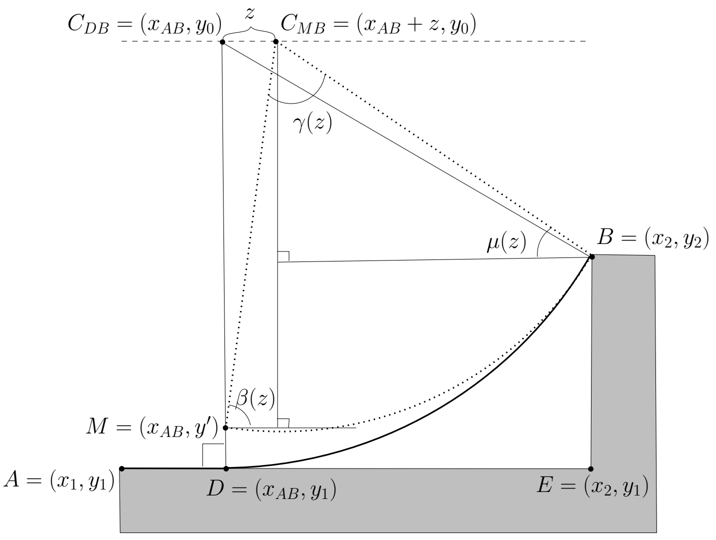


Figure 3.4: The optimal trajectory for a "one step" obstacle; T_{AB} hits L_{AE}

Lemma 3.2.1. *For three consecutive points $A = (x_1, y_1)$, $E = (x_2, y_1)$ and $B = (x_2, y_2)$ from S, such that $y_1 < y_2$, the time-optimal trajectory from A to B with obstacles is defined in the following way:*

a) if C_{AB} is to the left of A then T_{AB} is the time-optimal trajectory,

b) if C_{AB} is to the right of A then the time-optimal trajectory is a combination of the straight horizontal line segment L_{AD} and the circle segment T_{DB}, where D and C_{DB} determine a vertical line. In particular,

$$D = (x_2 - \sqrt{(y_2 - y_1)(2y_0 - y_1 - y_2)}, y_1). \tag{3.10}$$

Proof. a) Follows directly from Theorem 3.1.2, as T_{AB} does not hit the horizontal and vertical obstacles L_{AE} and L_{EB}.

b) T_{AB} hits L_{AE} as C_{AB} is to the right of A. Let us find the first point $D = (x, y_1)$ to the right of A such that T_{DB} does not hit the obstacle L_{AE}. The circle does not hit L_{AE} for the first time when L_{AE} is the tangent line of T_{DB}. So the x-coordinate of the circle center C_{DB} and D are the same. Hence, $R_{DB} = d(C_{DB}, D) = d(C_{DB}, B)$ and C_{DB} has coordinates (x, y_0). We have the following equation:

$$\begin{aligned}(x-x)^2 &+ (y_0-y_1)^2 = (x_2-x)^2 + (y_0-y_2)^2, \text{ thus,}\\ (x-x_2)^2 &= (y_2-y_1)(2y_0-y_1-y_2) \geq 0, \text{ as } y_0 \geq y_2 > y_1,\\ x &= x_2 - \sqrt{(y_2-y_1)(2y_0-y_1-y_2)}, \text{ as } x \leq x_2.\end{aligned}$$

Here and below we denote the found value of x for A and B as x_{AB}. We show now that the constructed trajectory L_{AD} plus T_{DB} is time-optimal with obstacles. Suppose there exists another trajectory T such that $t_T < t_{L_{AD}} + t_{T_{DB}}$. T has to cross the line $x = x_{AB}$ at some altitude above y_1 to avoid the obstacle L_{AE}, for example at point $M = (x_{AB}, y')$, $y' \in [y_1, y_0]$, see Figure 3.4. M splits the trajectory T into two parts:

- AM such that $t_{AM} \geq t_{L_{AD}}$, the proof is the same as the proof of Lemma 3.1.1,
- MB which is unknown, but the best we can do is to go from M to B along T_{MB}. Let us show that $t_{T_{MB}} \geq t_{T_{DB}}$.

  The position of the point C_{MB} is a one-to-one function of the position of the point M. For the simplicity we consider z as a parameter, where $C_{MB} = (x_{AB} + z, y_0)$. By Theorem 3.1.2 the time needed to go along T_{MB} is given by the following integral:

$$t_{T_{MB}}(z) = \frac{1}{q}\int_0^{\gamma(z)} \frac{d\alpha}{sin(\alpha + \beta(z))}.$$

  We show that the derivative of $t_{T_{MB}}(z)$ is positive and thus the function $t_{T_{MB}}(z)$ is monotonically increasing:

$$\begin{aligned} q\frac{dt_{T_{MB}}(z)}{dz} &= \int_0^{\gamma(z)} \frac{-\cos(\alpha+\beta(z))\beta'(z)d\alpha}{\sin^2(\alpha+\beta(z))} + \frac{\gamma'(z)}{\sin(\gamma+\beta)(z)} = \\ &= -\beta'(z)\int_{\sin\,\beta(z)}^{\sin(\beta(z)+\gamma(z))} \frac{d\tau}{\tau^2} + \frac{\gamma'(z)}{\sin(\gamma(z)+\beta(z))} = \\ &= \frac{\beta'(z)+\gamma'(z)}{\sin(\gamma(z)+\beta(z))} - \frac{\beta'(z)}{\sin\,\beta(z)} = \\ &= -\left[\frac{\beta'(z)}{\sin\,\beta(z)} + \frac{\mu'(z)}{\sin\,\mu(z)}\right],\end{aligned}$$

where $\mu(z) = \pi - \gamma(z) - \beta(z)$ is the angle between $C_{MB}B$ and the horizontal axis. Moreover, $R(z) = \sqrt{(x_{AB} + z - x_2)^2 + (y_0 - y_2)^2}$, and then $\cos\beta(z) = \frac{z}{R(z)}$, $\sin\beta(z) = \frac{\sqrt{R^2(z)-z^2}}{R(z)}$, $\cos\mu(z) = \frac{x_2 - x_{AB} - z}{R(z)}$, $\sin\mu(z) = \frac{y_0 - y_2}{R(z)}$. Hence we obtain,

$$q\frac{dt_{T_{MB}}(z)}{dz} = -\left[\frac{\beta'(z)}{sin\,\beta(z)} + \frac{\mu'(z)}{sin\,\mu(z)}\right] = \frac{z(x_2 - x_{AB})}{R(z)(R^2(z) - z^2)} \geq 0,$$

as the denominator is positive and $x_2 > x_{AB}$, so the time increases with the growth of z. $\frac{d}{dz}(qt_{T_{MB}}(z)) = 0$ if and only if $z = 0$, which means that $C_{MB} = C_{DB}$. We can conclude that $t_{T_{DB}}$ is smaller than t_{MB} for any trajectory MB.

So, the constructed trajectory L_{AD} plus T_{DB} is the time-optimal for one step as

$$t_{L_{AD}} + t_{T_{DB}} \leq t_{AM} + t_{T_{MB}} \leq t_T.$$

□

We observe that by Lemma 3.2.1 for any pair of points $A = (x_1, y_1)$, $B = (x_2, y_2)$ from S, it is sufficient to introduce at most one potential breakpoint $D = (x_{AB}, y_1)$ on the horizontal line $y = y_1$, if it is impossible to go along T_{AB} and it is possible to go along L_{AD} and T_{DB}. If there are other obstacles between A and B and the new trajectory hits some obstacle, then the time-optimal trajectory with rectilinear obstacles from A to B passes some other potential breakpoint E between them. This fact is stated in Lemma 3.2.5. Hence, all necessary potential breakpoints are determined by Lemma 3.2.1.

The main ingredients to prove Lemma 3.2.5 are the following propositions:

Proposition 3.2.2. *For any three points $A = (x_1, y_1), B = (x_2, y_2)$ and $M = (x_3, y_3)$ such that $x_1 < x_3 < x_2$ and $y_3 < y_1,\ y_3 < y_2$, T_{AB} lies above T_{AM} and T_{MB} on the interval $[x_1, x_2]$.*

Proof. Denote the x-coordinates of the centers of the circle segments T_{AM}, T_{MB} and T_{AB} as x_0^{AM}, x_0^{MB} and x_0^{AB} respectively. Then $x_0^{MB} \leq x_0^{AB} \leq x_0^{AM}$ as $x_0^{AB} = \lambda x_0^{MB} + (1 - \lambda)x_0^{AM}$ for $\lambda = \frac{x_2 - x_3}{x_2 - x_1} \in [0, 1]$.

The circle segment T_{AB} is given by the equation $(x - x_0^{AB})^2 + (y - y_0)^2 = R^2_{AB}$, where $R^2_{AB} = (x_0^{AB} - x_1)^2 + (y_0 - y_1)^2$, so $y = y_0 - \sqrt{R^2_{AB} - (x - x_0^{AB})^2}$. Similarly, the circle segment T_{AM} is given by the equation $y = y_0 - \sqrt{R^2_{AM} - (x - x_0^{AM})^2}$, where $R^2_{AM} = (x_0^{AM} - x_1)^2 + (y_0 - y_1)^2$. The circle segment T_{AB} is above the circle segment T_{AM} if $y_0 - \sqrt{R^2_{AB} - (x - x_0^{AB})^2}$ is bigger than $y_0 - \sqrt{R^2_{AM} - (x - x_0^{AM})^2}$ for any $x \in [x_1, x_2]$. To verify this fact, it is enough to compare the expressions under the radical:

$$R^2_{AB} - (x - x_0^{AB})^2 - (R^2_{AM} - (x - x_0^{AM})^2) = 2(x_0^{AB} - x_0^{AM})(x - x_1) \leq 0.$$

So $\sqrt{R_{AB}^2 - (x - x_0^{AB})^2} \leq \sqrt{R_{AM}^2 - (x - x_0^{AM})^2}$ for any $x \in [x_1, x_2]$, hence,

$$y_0 - \sqrt{R_{AB}^2 - (x - x_0^{AB})^2} \geq y_0 - \sqrt{R_{AM}^2 - (x - x_0^{AM})^2}.$$

Therefore, T_{AB} is above T_{AM}. The proof that T_{AB} is above T_{MB} is similar. □

Proposition 3.2.3. *For any three consecutive points $A = (x_1, y_1), C = (x_2, y_1)$ and $B = (x_2, y_2)$ such that $x_1 < x_2$ and T_{AB} hits L_{AC}, the time-optimal trajectory lies above T_{AB} on the interval $[x_1, x_2]$.*

Proof. We consider $x_1 = 0$ and $D = (\Delta x, y_1)$ without loss of generality, where Δx is determined by Equation (3.10) as in this case L_{AD} plus T_{DB} is the time-optimal trajectory with one step obstacle. Furthermore, $\Delta x = x_0^{DB}$. Then, by Equation (3.7)

$$\begin{aligned} x_0^{AB} &= \frac{-2(y_2 - y_1)y_0 - y_1^2 + x_2^2 + y_2^2}{2x_2}, \\ x_0^{DB} &= \Delta x = \frac{-2(y_2 - y_1)y_0 - y_1^2 + x_2^2 + y_2^2}{2(x_2 - \Delta x)} + \frac{(\Delta x)^2}{2(\Delta x - x_2)}. \end{aligned}$$

We conclude that

$$\begin{aligned} 2x_0^{DB}(x_2 - \Delta x) + (\Delta x)^2 &= 2x_2 x_0^{AB}, \text{ hence,} \\ x_0^{DB} - x_0^{AB} = \Delta x - x_0^{AB} &= \frac{(\Delta x)^2}{2x_2} \geq 0. \end{aligned}$$

Therefore, $x_0^{DB} \geq x_0^{AB}$, so the circle segment T_{DB} is above T_{AB} for any $x \in [x_1, x_2]$ (the proof of this fact is similar to the proof of Proposition 3.2.2). □

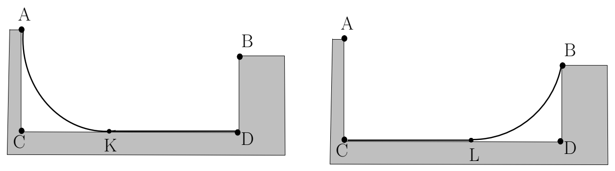


Figure 3.5: The time-optimal trajectory for the unique horizontal obstacle between A and B

We conclude the set of propositions with the so-called "ground level case", when the unique obstacle between A and B is a horizontal line.

Proposition 3.2.4. *Given four consecutive points $A = (x_1, y_1), C = (x_1, y_2), D = (x_2, y_2)$ and $B = (x_2, y_3)$ from S such that $x_1 < x_2,\ y_2 < y_1,\ y_2 < y_3$ and T_{AB} hits L_{CD}, the time-optimal trajectory with obstacles is the combination of the circle segment T_{AK}, straight line L_{KL} and circle segment T_{LB}, where K and L are the points of intersection of T_{AD} and T_{CB} with L_{CD} respectively.*

Proof. We consider the triplets $A,\ C,\ D$ and $C,\ D,\ B$ as the input to Lemma 3.2.1 (see Figure 3.5). T_{AK} does not intersect T_{LB} by Proposition 3.2.2, as T_{AB} hits L_{CD}. □

Finally, we present Lemma 3.2.5. The proof is the extensive case study for the obstacle patterns between two points A and B.

Lemma 3.2.5. *If the time-optimal trajectory T_{AB} between any two potential breakpoints $A = (x_1, y_1), B = (x_2, y_2),\ \ x_1 < x_2$, determined by Lemma 3.2.1 hits the obstacles, then there exists a potential breakpoint C between A and B such that the time-optimal trajectory with obstacles from A to B passes C.*

Proof. Assume the statement is not correct, hence the trajectory hits some obstacle and there exists a time-optimal trajectory T with obstacles from A to B such that it does not hit any obstacle and does not pass any potential breakpoint.

We split the proof into five parts depending on the type of "terrain" between A and B.

1) *The obstacles between A and B are monotonically increasing (or decreasing), see Figure 3.6.*

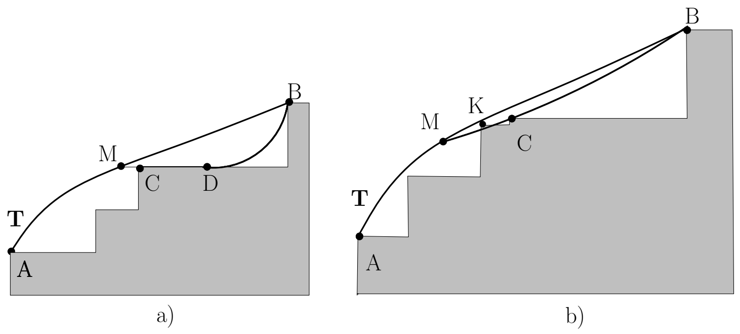


Figure 3.6: Monotonic obstacle case

First we notice that T is convex, otherwise we can always locally shortcut it with the straight line in such a way that it still does not pass any potential breakpoint; but the straight line is always better than any trajectory above it by Lemma 3.1.1, so T is not time-optimal.

We assume without loss of generality that $y_1 < y_2$. Let us consider the last obstacle $C = (x_3, y_3)$ on the way along T, T does not pass C by assumption. By Lemma 3.2.1 the time-optimal trajectory from C to B has one of two shapes:

- it is the combination of the straight line L_{CD} and the circle segment T_{DB}, see Figure 3.6a). The intersection of the line $y = y_3$ and the trajectory T

is point M. The time-optimal trajectory from M to B is the combination of the straight line L_{MD} and the circle segment T_{DB} by Lemma 3.2.1b). Then the following combined trajectory is better than T: T before the point M, straight line L_{MD} and circle segment T_{DB}, and it passes the potential breakpoint C, a contradiction.

- it is T_{CB}, see Figure 3.6b). M is the point of intersection of T and T_{CB}. If T_{MB} does not hit any obstacle before crossing T or it just crosses the obstacle in the potential breakpoint K then the trajectory T between A and M plus T_{MB} is better than T and it passes the potential breakpoint C, a contradiction.

  If T_{MB} hits the obstacle which is determined by the potential breakpoint K then we consider T_{KB} (or L_{KD_K} and T_{D_KB}) by Lemma 3.2.1a) (or b)) and so on. The number of obstacles is finite so we can find the "last" obstacle L such that the time-optimal trajectory from Lemma 3.2.1 does not hit this obstacle. Any new circle segment or combination of the line and circle segment lies higher than the previous one by induction, so it does not hit any obstacle to the right of the considered potential breakpoint. We use the time-optimal trajectory which is determined by the potential breakpoint L and B to shortcut T as in the previous case, a contradiction.

2) *There is an obstacle between A and B which is higher than A and B, see Figure 3.7.* By Lemma 3.1.1 we can always shortcut T with the line L_{KM} which passes 2 potential breakpoints, a contradiction.

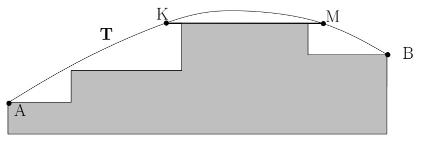


Figure 3.7: Concave obstacle case

3) *The obstacles between A and B is a convex rectilinear function, see Figure 3.8.* Similar to case 1) the trajectory T is convex. Denote the lowest point of the trajectory T as point $M = (x_M, y_M)$. We consider two cases:

   – *All the obstacles which are hit by T_{AB} are lower than M, see Figure 3.8a).* We introduce a new instance: the horizonal line $y = y'$ at the altitude of the highest obstacle $K = (x', y')$, hit by T_{AB}. T is a valid trajectory for the new instance. The time-optimal trajectory T' for the new instance by Proposition 3.2.4 is the circle segment T_{AK_1}, line segment $L_{K_1K_2}$ and circle segment T_{K_2B}, where K_1, K_2 belong to the line $y = y'$. By Proposition 3.2.3

this trajectory does not hit any obstacle of the original instance, so it is a valid trajectory for the original instance. T' is time-optimal for the new instance, hence it is better than T. So, T' is also better than T for the original instance, a contradiction.

- *There is at least one obstacle $K = (x', y')$, which is hit by T_{AB}, and it is higher than M, see* ***Figure 3.8b)****.* We introduce a new instance: the horizontal line $y = y_M$. If either T_{AM} or T_{MB} hits the obstacles then we are in the settings of the case 1) for one of the segments T_{AM} or T_{MB}, hence there is a trajectory T' which is better than T, and it passes one of the potential breakpoints.

  So we only consider the case when T_{AM} and T_{MB} does not hit any obstacle. It means that T is just a combination of T_{AM} and T_{MB}. By Proposition 3.2.2 T_{AB} is above T, so T_{AB} also does not hit any obstacle, a contradiction.

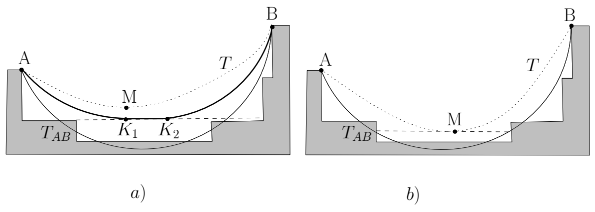


Figure 3.8: Convex obstacle case

4) *There is an obstacle between A and B which is higher than A, lower than B and it is the point of non-monotonicity, see* ***Figure 3.9****.* If the trajectory T "goes down" after this obstacle, see Figure 3.9a), we can shortcut it as in case 2), a contradiction. If it stays higher, see Figure 3.9b), then we can introduce a new equivalent monotonic instance where the non-monotonic part is replaced by a horizontal dashed line. So we are in the settings of case 1).

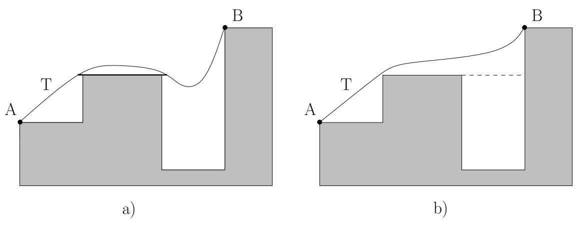


Figure 3.9: Non-monotonic obstacle case, the obstacle is higher than A

5) *The obstacles between A and B are not presented by a convex rectilinear function, and all the obstacles are lower than A and B, see* ***Figure 3.10****.* The obstacle curve

is not convex, so there exists an obstacle which is concave. If T “goes down” on both sides of this obstacle we can shortcut is as in case 2), see Figure 3.10a), a contradiction. If T stays higher at least on one side, see Figure 3.10b), then we can introduce a new equivalent convex instance with a dashed line as in case 4), so we are in the settings of the case 3).

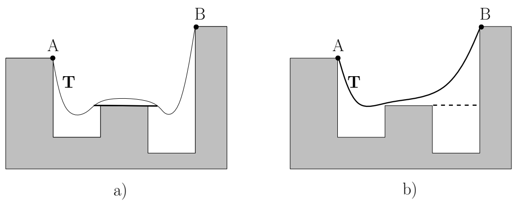


Figure 3.10: Non-monotonic obstacle case, the obstacle is lower than A and B

Finally we conclude that if the time-optimal trajectory between two potential breakpoints A and B hits an obstacle then there is a potential breakpoint C between A and B such that the time-optimal trajectory from A to B passes C. □

Now the problem is reduced to the problem of finding a shortest path in a weighted graph. All the potential breakpoints are the vertices of the graph, the number of vertices is at most $n + n^2$ =O(n^2) (where n is the number of obstacles given by the set S). There is an edge between two vertices if the time-optimal trajectory between corresponding potential breakpoints, given by Lemma 3.2.1, does not hit any obstacle. The time needed to go along this trajectory denotes the weight of the edge. Lemma 3.2.5 guarantees that the graph is connected and the time-optimal trajectory corresponds to the shortest path in the weighted graph. This problem is solvable in time O($|V|^2$), see, e.g. [74]. Hence, we have O(n^4)-time algorithm for 2D basic helicopter problem with rectilinear obstacles.

3.3 Conclusions and open problems

In this chapter we have addressed the problem of determining a time-optimal helicopter trajectory between two points in two-dimensional space, where the speed of the helicopter depends linearly on the altitude. We have characterized the motion primitives for the basic helicopter problem. We used these time-optimal trajectories to solve the basic helicopter problem with rectilinear obstacles in the time polynomial in the number of obstacles. It is still an open question whether it is possible to obtain a polynomial time algorithm for an arbitrary continuous obstacle curve.

Chapter 4

General Helicopter Problem without Obstacles

In this chapter we naturally generalize the problem considered in the previous chapter to the problem with a piecewise linear concave velocity function with a finite number n of breakpoints, see [8] for motivation. We call this problem a *general helicopter problem*. We derive a set of motion primitives for the general helicopter problem. However, the considered setting does not allow "closed" and easy to use analytical representation of the motion primitives as it was in Chapter 3. For this reason, we assume throughout this chapter that no obstacles and no ground level are present.

We obtain a function of n variables which gives the time needed to travel from one point to another along a potential time-optimal trajectory. The rest of the chapter is devoted to a deep analysis of this function in order to determine the time-optimal trajectory. The methodology is based on Fermat's Theorem 2.1.2, see Chapter 2; namely we use the claim that a continuous function on a compact domain reaches it's minimum in one of the points: on the boundary of the domain or in the interior of the domain where the derivative is not determined or equal to zero. Is is impossible to use variety of convex optimization methods like Newton's method as we show that the function is not convex.

We reduce the general helicopter problem without obstacles to a system of fixed degree multivariable polynomial equations. If a number of breakpoints in the piecewise linear velocity function is a constant, the resulting system has a fixed number of variables and it can be solved in finite number of steps by algebraic elimination theory; see, e.g., [33]. The advantage of this approach is that the heavy techniques of elimination theory should be used only once in a symbolic form as a preprocessing step. It provides us with a high degree polynomial equation at one variable with the coefficients depend on a set of the problem's parameters including all air characteristics, a helicopter's technical parameters and a starting and finishing point of a trajectory. Later on a set of parameters together with a desired level of precision can be given and some quick method of solving an univariable equation is used, like Jenkins-Traub algorithm, see [77]. It gives us a promising method of solving some practical problems like autopilot flights, for example. We assume a computational model with infinite precision real arithmetics to avoid the problem of comparison the lengths of two paths, see, e.g., Mitchell and Sharir [71].

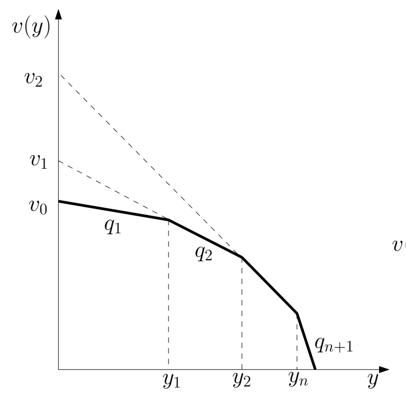
The chapter is organized as following: Section 4.1 contains the formulation of the problem; here, we introduce a set of variables in order to reduce the problem to the basic helicopter problem without obstacles and to introduce the time function. Section 4.2 is devoted to a search of the stationary points of the derived time function. In Section 4.3 we study the boundary of the domain and search the local extreme points. In all these sections we assume that the starting point is “close” to the ground level and the end point is “close” to the maximum altitude. In the next section we generalize the position of the starting and the end points of the trajectory. Section 4.5 contains the motivation of the approach described above; namely it shows that the time function is not convex. Hence, the standard techniques of the convex optimization like Newton’s method are not applicable here. The last section contains conclusions and some open problems.

4.1 Problem definition

In this section we consider the general helicopter problem where the velocity function is a piece-wise linear concave function in altitude with finite number of breakpoints $y_1, \dots, y_n$, see Figure 4.1. Let $v_i, i \in I = \{0, \dots, n\}$ denote the intercept of the velocity function between the altitudes $y = y_i$ and $y = y_{i+1}$. By concavity of the function we have that $q_1 \leq q_2 \leq \dots \leq q_{n+1}$. This is a reasonable assumption as the helicopter looses the speed quicker on high altitudes. Let us define the altitude layers L_i, $i \in I$, in the following natural way: $L_i = \{y \mid y_i \leq y \leq y_{i+1}\}, i \in I \setminus \{0, n\}$, $L_0 = \{y \mid y \leq y_1\}$ and $L_n = \{y \mid y \geq y_n\}$. Therefore, the velocity function is given as follows:

Consider the general helicopter problem where the helicopter must fly from $A = (x_A, y_0)$ to $B = (x_B, y_{n+1})$. We assume, that $x_A < x_B$ and $y_0 \in L_0,\ y_{n+1} \in L_n$. Section 4.4 contains the adjustment to the arbitrary choice of the layer for the starting point.

Given setting yields several natural restrictions. First, we observe that the time-optimal trajectory from A to B is the set of circle and linear segments, as within one layer $L_i,\ i \in \{0, \dots, n\}$, it is either the circle segment or the linear segment, see Chapter 3, Section 3.1. Second, for any $y \geq y_1$ the time-optimal trajectory is the monotonically increasing function. The reason is as follows. Assume that a non-increasing trajectory T_{KB} is determined as the time-optimal trajectory within the



$$v(y) = \begin{cases} v_0 - q_1 y & : y \in L_0, \\ v_1 - q_2 y & : y \in L_1, \\ \dots & \dots \\ \max\{v_n - q_{n+1} y, 0\} & : y \in L_n. \end{cases}$$

Figure 4.1: Piece-wise linear velocity function

layer L_1, see Figure 4.2. Then the piece T_{KL} of this trajectory, which belongs to the layer L_0, is not optimal anymore. It is better to follow T_{AL} within the layer L_0 and then continue with T_{LB}, a contradiction.

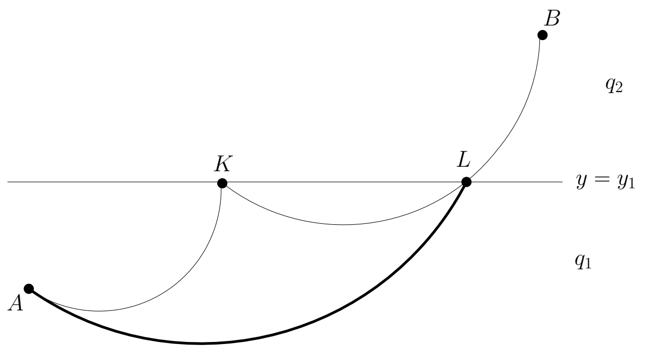


Figure 4.2: Monotonically increasing circle segments when $y \geq y_1$. The trajectory T_{AL} plus T_{LB} is better than T_{AK} plus T_{KB}

As the time-optimal trajectory is the set of circle and line segments and the time function is continuous (see Chapter 3) than the time needed to go along this trajectory can be calculated as the sum of times to go along the circle segments in layers L_i's, $i \in I$.

For every layer $L_i,\ i \in I$ we introduce the set of parameters, see Figure 4.3:

- $y_0^{i+1} = \frac{v_i}{q_{i+1}}$ is the altitude of the time-optimal circle segment center within this layer,
- $a_{i+1} = \sqrt{(2y_0^{i+1} - y_{i+1} - y_i)(y_{i+1} - y_i)}$, as is determined in Lemma 3.2.1. It is the maximal horizontal distance between two points on lines $y = y_i$ and $y = y_{i+1}$ such that the time-optimal circle segment is a monotonically increasing function,

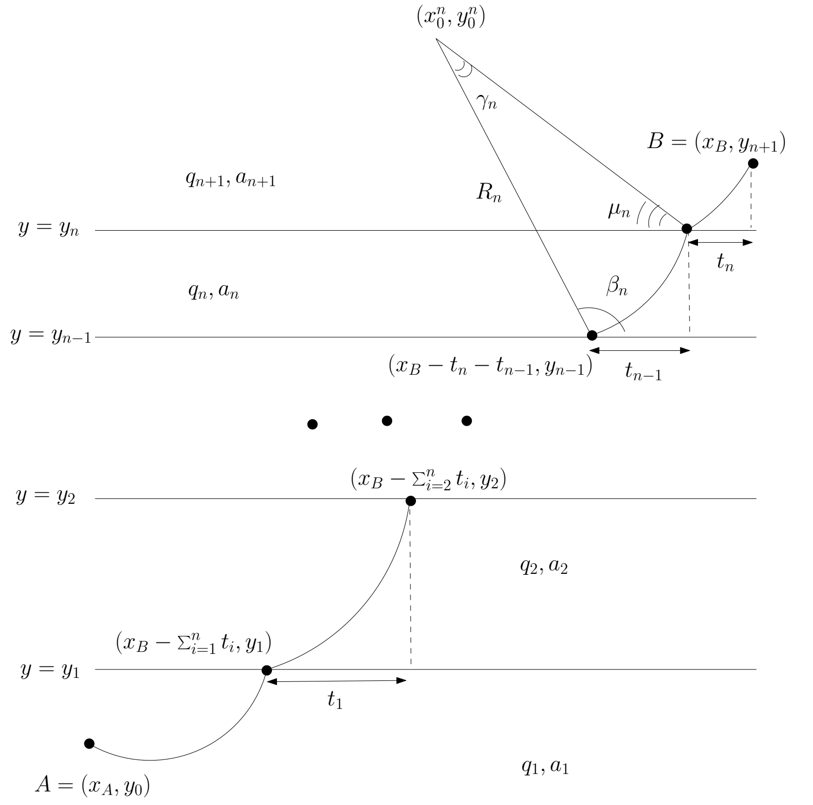


Figure 4.3: Time-optimal trajectory is the set of circle segments; $x_0^i, \beta_i, \mu_i, \gamma_i$ and R_i are functions at $t_1, \dots, t_n$

and a set of variables and functions:

- t_i, such that for any i greater than zero t_i is the horizontal distance between the points of intersection the lines $y = y_i$ and $y = y_{i+1}$ by the time-optimal trajectory from A to B, so that the coordinates of intersection are $(x_B - \sum_{i=k}^n t_i, y_k), k \in \{1, \dots, n\}$. The values of t_i's are strictly positive, as otherwise we have a vertical line segment, and $t_i \leq a_{i+1}, i > 0$, as the circle segment should be a monotonically increasing function for $i > 0$. We introduce the redundant variable t_0 such that $x_B - \sum_{i=0}^n t_i = x_A$, where t_0 is the horizontal distance between point A and the point of intersection of the time-optimal trajectory from A to B with the line $y = y_1$. It is strictly positive, as otherwise we go further to the left than point A, which is not optimal. It can be bigger than a_1, as within the layer L_0 the time-optimal trajectory is not necessary the monotonically increasing function,

- $x_0^{i+1}(t_1, \dots, t_n)$ is the x-coordinate of the time-optimal circle segment center within this layer,

- $R_{i+1}(t_1, \dots, t_n)$ is the radius of the time-optimal circle segment within this layer,

- $\beta_{i+1}(t_1, \dots, t_n)$ and $\gamma_{i+1}(t_1, \dots, t_n)$ are the angles defined in Theorem 3.1.2; we also introduce the angle $\mu_{i+1}(t_1, \dots, t_n)$ between CB and x-axis in terms of Figure 3.2, in particular, $\mu_{i+1} = \pi - \gamma_{i+1} - \beta_{i+1}$.

Now, we can write down the time needed to go alone this trajectory as the sum of the times to go alone the time-optimal trajectories within one layer L_i, see Chapter 3:

$$t(t_1, \dots, t_n) = \sum_{k=0}^{n} \frac{1}{q_{k+1}} \int_0^{\gamma_{k+1}(t_1,\dots,t_n)} \frac{d\alpha}{sin(\alpha + \beta_{k+1}(t_1, \dots, t_n))},$$

where $t_i \in [0, a_{i+1}], i \in \{1, \dots, n\}$.

4.2 Stationary points of the time function

In this and the following sections we minimize the time function in the standard way. The time function is continuous as a sum of continuous functions, thus by Extreme Value Theorem 2.1.1 $t(t_1, \dots, t_n)$ must attain its minimum value at the domain $[0, a_2] \times \dots \times [0, a_{n+1}]$. We introduce the notation $\bar{t} = (t_1, \dots, t_n)$ for the simplicity.

To minimize $t(\bar{t})$ we have to find the stationary points in the interior of the function domain and the minimum value on the boundary of the domain. This section contains the necessary conditions for the stationary points in $(0, a_2) \times \dots \times (0, a_{n+1})$. In the next section is devoted to the study of the boundary, where at least one of t_i's is equal to zero or a_{i+1}.

To find the stationary points we have to find the partial derivatives of $t(\bar{t})$ with respect to $t_i, i \in \{1, \dots, n\}$ and equate them with zero. Direct differentiation leads to the following results, see Lemma 3.2.1 for details:

$$\frac{\partial t(\bar{t})}{\partial t_i} = -\sum_{k=1}^{n+1} \frac{1}{q_k} \left[\frac{\partial \beta_k(\bar{t})}{\partial t_i} \frac{1}{\sin\ \beta_k(\bar{t})} + \frac{\partial \mu_k(\bar{t})}{\partial t_i} \frac{1}{\sin\ \mu_k(\bar{t})} \right] = 0.$$

We find $\beta_k(\bar{t})$ and $\mu_k(\bar{t})$, $k \geq 2$, using the formulae from Chapter 3, as all the derivatives in this case can be obtained in the similar way. The case $k = 1$ differs in the number of variables and it is considered later. Within the layer $L_{k-1}, k \geq 2$, the time-optimal trajectory goes from the point $(x_B - \sum_{i=k-1}^{n} t_i, y_{k-1})$ to the point $(x_B - \sum_{i=k}^{n} t_i, y_k)$, so from Equations (3.7)–(3.8) and some formulae for the trigonometrical functions:

$$\begin{aligned}
x_0^k(\bar{t}) &= -\frac{a_k^2}{2t_{k-1}} + x_B - \sum_{i=k}^{n} t_i - \frac{t_{k-1}}{2}, \text{ then}\\
R_k(\bar{t}) &= \sqrt{\left(\frac{a_k^2}{2t_{k-1}} + \frac{t_{k-1}}{2}\right)^2 + (y_0^k - y_k)^2} = R_k(t_{k-1}), \text{ and}\\
\sin\ \beta_k(\bar{t}) &= \frac{y_0^k - y_{k-1}}{R_k(t_{k-1})},\\
\cos\ \beta_k(\bar{t}) &= \frac{x_0^k(t_1,\ldots,t_n) - (x_B - \sum_{i=k-1}^{n} t_i)}{R_k(t_{k-1})} = -\frac{a_k^2 - t_{k-1}^2}{2t_{k-1}R_k(t_{k-1})},\\
\sin\ \mu_k(\bar{t}) &= \frac{y_0^k - y_k}{R_k(t_{k-1})},\\
\cos\ \mu_k(\bar{t}) &= \frac{x_B - \sum_{i=k}^{n} t_i - x_0^k(t_1,\ldots,t_n)}{R_k(t_{k-1})} = \frac{a_k^2 + t_{k-1}^2}{2t_{k-1}R_k(t_{k-1})}.
\end{aligned}$$

Thus, we see that $R_k,\ \beta_k$ and μ_k are functions of only t_{k-1}, so for any other $i, i \neq (k-1)$, the partial derivatives of β_k and μ_k with respect to t_i are equal to zero. After differentiation and some simplifications we conclude that

$$\frac{\beta_k'(t_{k-1})}{sin\ \beta_k(t_{k-1})} + \frac{\mu_k'(t_{k-1})}{sin\ \mu_k(t_{k-1})} = -\frac{1}{R_k(t_{k-1})},\ k = 2,\ldots,n+1.$$

We find $\beta_1(\bar{t})$ and $\mu_1(\bar{t})$ in the similar way. The start point of the time-optimal trajectory within the layer L_0 is $A = (x_A, y_0)$ and the end point is $(x_B - \sum_{k=1}^{n} t_i, y_1)$. The only difference is that these functions depend on all n variables $t_1,\ldots,t_n$.

We find by differentiation that:

$$\frac{\partial\mu_1}{\partial t_i}\frac{1}{\sin\ \mu_1(\bar{t})} + \frac{\partial\beta_1}{\partial t_i}\frac{1}{\sin\ \beta_1(\bar{t})} = \frac{1}{R_1(\bar{t})},\ i = 1,\ldots,n.$$

Hence, the stationary point of the function $t(\bar{t})$ satisfies the following set of conditions:

$$\frac{\partial t(\bar{t})}{\partial t_k} = -\left(\frac{1}{q_1 R_1(\bar{t})}\right) - \left(-\frac{1}{q_{k+1}R_{k+1}(t_k)}\right) = 0,\ k = 1,\ldots,n, \text{ or equivalently}$$

$$q_1 R_1(t_1,\ldots,t_n) = q_2 R_2(t_1) = \ldots = q_{n+1}R_{n+1}(t_n), \tag{4.1}$$

where $t_i \in (0, a_{i+1})$ and $x_B - x_A - \sum_{i=1}^{n} t_i = t_0 > 0$.

It is possible to simplify the set of Equations (4.1) further: consider every consecutive pair of expressions k and $k+1$, $k = 1,\ldots,n$, and use the fact that the point $(x_B - \sum_{i=k}^{n} t_i, y_k)$ belongs at the same time to the circles with radii $R_k(t_{k-1})$ and $R_{k+1}(t_k)$

as the time-optimal trajectory has to be continuous. We use the redundant variable t_0 to write down the system in the uniform way:

$$q_{k+1}(a_{k+1}^2 - t_k^2)t_{k-1} = q_k(a_k^2 + t_{k-1}^2)t_k, \; k = 1, \dots, n, \tag{4.2}$$

$$\sum_{i=0}^{n} t_i = x_B - x_A, \tag{4.3}$$

$$t_i \in (0, a_{i+1}), \; i > 0, \qquad t_0 > 0. \tag{4.4}$$

Let us show that at most one solution of the system of Equations (4.1) on the domain $\Pi_{i=1}^n(0, a_{i+1})$ exists, hence the system (4.2)—(4.4) also has at most one solution. Suppose there exists a solution $(t_1^*, \dots, t_n^*)$ of the system (4.1) with the corresponding trajectory T, see Figure 4.4. Assume that there is another solution $(t_1', \dots, t_n')$ with the corresponding trajectory T'. The circle segments of both trajectories satisfy the system of Equations (4.1), so

$$\begin{aligned} q_1 R_1(t_1^*, \dots, t_n^*) &= q_2 R_2(t_1^*) = \dots = q_{n+1} R_{n+1}(t_n^*), \text{ and} \\ q_1 R_1(t_1', \dots, t_n') &= q_2 R_2(t_1') = \dots = q_{n+1} R_{n+1}(t_n'). \end{aligned}$$

We denote $R_i(t_1^*, \dots, t_n^*)$ as R_i^* and $R_i(t_1', \dots, t_n')$ as R_i'.

Assume that T' is completely above T and they intersect only in the end points A and B. K^* and L^* are the points where T intersects $y = y_1$ and $y = y_n$ respectively, K' and L' are the points where T' intersects $y = y_1$ and $y = y_n$ respectively. K' is to the left of K^* and the left end A of the trajectories T and T' is fixed, thus, $R_1^* < R_1'$. L' is to the left of L^* and the right end B of T and T' is fixed, thus, $R_{n+1}^* > R_{n+1}'$. Both T and T' represent the solution of the system (4.1) so

$$q_{n+1}R_{n+1}' < q_{n+1}R_{n+1}^* = \dots = q_1 R_1^* < q_1 R_1' = \dots = q_{n+1} R_{n+1}',$$

a contradiction. All other cases, like T' is completely below T or they intersect in some intermediate point $M \neq B$, are proven in the same way. Therefore, at most one solution of the system (4.1) exists.

Furthermore, it is easy to check that if $t_k \in (0, a_{k+1})$ is known then there exists at most one value of $t_{k-1} \in (0, a_k)$, because two possible values of t_{k-1} from Equation (4.2) are:

$$t_{k-1}^{1,2} = \frac{q_{k+1}(a_{k+1}^2 - t_k^2) \pm \sqrt{q_{k+1}^2(a_{k+1}^2 - t_k^2)^2 - 4q_k^2 a_k^2 t_k^2}}{2q_k t_k}, \tag{4.5}$$

if the discriminant $q_{k+1}^2(a_{k+1}^2 - t_k^2)^2 - 4q_k^2 a_k^2 t_k^2$ is nonnegative. $t_{k-1} < a_k$, so only the solution with *minus* reminds, as otherwise the condition is broken. Hence, if the discriminant is nonnegative, there exists a unique solution $t_{k-1} = t_{k-1}(t_k)$. So, given

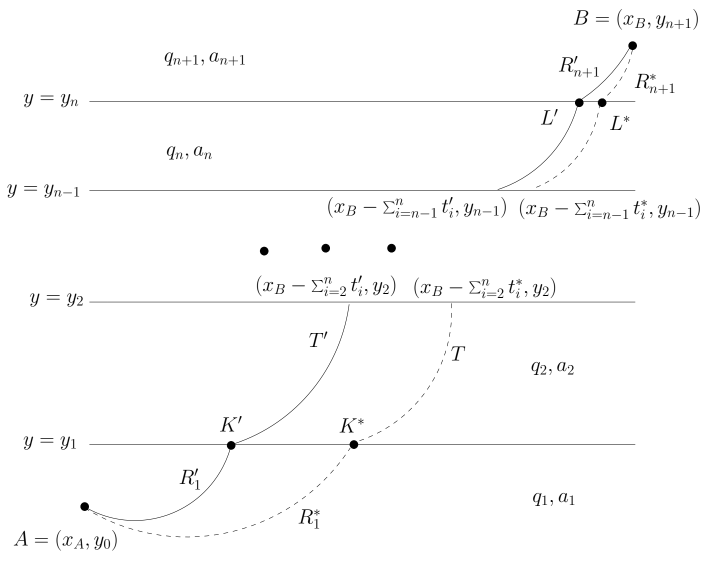


Figure 4.4: Uniqueness of the solution

t_n we can find $t_k, k = n-1, \ldots, 1$, using the next equation from the set. Therefore, either there exists a unique solution of the system or there is no solution, so the time-optimal trajectory which crosses $(x_B - t_n, y_n)$ and which is the set of circle segments does not exist.

Now, we have to solve the system of polynomial equations (4.2)—(4.4). We use the tools of computational algebraic geometry and commutative algebra to solve it. The brief introduction into the theory about polynomials of n variables $x_1, \ldots, x_n$ is given in Chapter 2, see also Cox [33].

In particular, the Elimination Theorem 2.2.1 guaranties that we can find a polynomial consequence of the system (4.2)—(4.4) in finitely many steps. It means that we will obtain the following Equation:

$$\sum_{i=0}^{m} t_n^i * C_i(y_0, \ldots, y_{n+1}, x_A, x_B, v_0, q_1, \ldots, q_{n+1}) = 0, \tag{4.6}$$

where $C_i(y_0, \ldots, q_{n+1})$ is the coefficient of t_n^i. These coefficients are constants as they depend only on the parameters of the problem. The nice thing about this approach is that the Elimination Theorem should be applied only once as a preprocessing step in symbolic form for any n. We will obtain Equation (4.6) where the coefficients are functions at parameters of the problem. Later on for every particular problem we just have to substitute these parameters with their corresponding values. Afterwards we solve the equation as it contains only t_n as a variable. There are numerical methods to find the roots of such an equation with high precision (for example, the Jenkins-Traub algorithm, see [77]). If there exists a root $t_n^* \in (0, a_{n+1})$ then we can find all other values of t_k^*, $k = n-1, \ldots, 1$ from Equation (4.5). Otherwise there is no time-optimal trajectory which is a set of circle segments and we have to find the minimum of the time-function on the boundary of its domain.

There are some practical problems with the method suggested above, see [68, 93]. Computing the roots of the univariate polynomial can be ill-conditioned for polynomials of degree greater than 14, 15, while Gröbner basis can contain as a consequence a polynomial (4.6) at one variable of high degree m. In fact, the polynomial consequence (4.6) is a crucial one. As an example, if the velocity function has only one breakpoint then the highest degree of the polynomial at t_n is $m = 3$ (hence, it is solvable analytically); for two breakpoints the highest degree is $m = 7$. So, if the number of breakpoints is small then the method is applicable.

Nevertheless, the proposed approach has some positive sights. First, the obtained system has at most one root in the domain as it is proved above. Therefore, the desired solution is separated from the rest which increases the stability of the search. Second, the system includes all the parameters of the helicopter as the parameters of the coefficients. Thus we have to apply elimination theory in the symbolic form only once, for example as a preprocessing step. We obtain the polynomial consequence of the system which depends only on one variable. Later on we substitute the coefficients of this equations with the values specific for every helicopter, medium properties and the flight starting and destination points. The solution of the particular problem is derived by some quick methods of solving the equations at one variable like already mentioned above Jenkins-Traub [77] algorithm with given level of precision. Therefore, the approach can be useful in creating the autopilot system for the helicopter as the most time-consuming operations are done beforehand.

4.3 Boundary search

In this section we briefly sketch the search of the potential time-optimal solution which contains circle and line segments and belongs to the boundary of the function domain. In fact, we reduce this case to the previous one, which contains only circle segments. $(t_1, \ldots, t_n)$ belongs to the boundary of the domain if there exists $t_i = 0$ or $t_i = a_{i+1}$, so we have

$$\sum_{i=1}^{n} \begin{pmatrix} n \\ i \end{pmatrix} \times 2^k \leq (2+1)^n = 3^n$$

combinations of possible sets of t_k's. We assume that the number of breakpoints of the velocity function n is constant, so 3^n is a constant factor.

Assume that only one t_k is equal to zero or a_{i+1}, then the time needed to go alone the trajectory is:

$$t(t_1,\dots,t_{k-1},t_{k+1},\dots t_n) = \sum_{\substack{i=1,\\ i\neq k+1}}^{n} \int_0^{\gamma_i(\bar{t})} \frac{d\alpha}{\sin(\alpha+\beta_i(\bar{t}))} + t_{const}, \tag{4.7}$$

where t_{const} is the time to go along the vertical line segment within the layer L_k from the altitude $y = y_k$ to the altitude $y = y_{k+1}$ if $t_k = 0$ and t_{const} is the time to go from the point (x, y_k) to the point $(x + a_{k+1}, y_{k+1})$ if $t_k = a_{k+1}$. It is a constant with respect to the variables $t_1, \dots, t_{k-1}, t_{k+1}, \dots, t_n$.

We find the stationary point conditions for Equation (4.7) in the similar way as in Section 4.2. The partial derivative of the term t_{const} with respect to any $t_i, i \neq k$, is equal to zero, as it is a constant. The functions $\beta_i(t_1,\dots,t_{k-1},t_{k+1},\dots,t_n)$ and $\mu_i(t_1,\dots,t_{k-1},t_{k+1},\dots,t_n)$, $i \neq (k+1)$, are still the functions at t_{k-1} only, as we do the same steps, replacing t_k with zero or a_{k+1}. β_{k+1} and μ_{k+1} are not determined any more. So we end up with the similar system of equations:

$$q_1R_1(t_1,\dots,t_{k-1},t_{k+1},\dots,t_n) = q_iR_i(t_{i-1}),\ i \in \{1,\dots,n\} \setminus \{k+1\}. \tag{4.8}$$

If more than one of the t_k's are equal to zero or a_{k+1}, then Equation (4.7) contains more constant factors, but the rest of the analysis remains the same. Finally, we choose the best trajectory with respect to time (determined by Equation (4.7)) among the trajectories generated by some set of t_k's which are equal to zero or a_{k+1}. The case of the stationary point in the interior of the domain will be included if we consider the empty set of t_k's which are equal to zero or a_{k+1}.

4.4 Arbitrary position of the starting point

In this section we assume that the point $A = (x_A, y_A)$ belongs to the layer $L_{k+1}, 0 \leq k \leq n$. Without loss of generality we still assume that the end point $B = (x_B, y_B)$ belongs to the highest level L_{n+1} as otherwise we restrict ourselves to the equivalent setting of bottom layers. We modify the system of equations (4.2)—(4.4) to solve this case. We consider the similar set of variables $t_0, \dots, t_n$, see Figure 4.5. The time-optimal trajectory may not reach the layer L_0, so we assume that the lowest point of the trajectory belongs to the layer $L_m, 0 \leq m \leq k+1$. We have to consider all of these possible m's, so we have $k + 1 \leq n + 1$ different cases where the number of velocity function breakpoints n is finite. We denote $\bar{t} = (t_0, t_m, \dots, t_n)$ similar to the previous sections.

Notice that the time-optimal trajectory is symmetric within bottom $(k + 1)$ layers. We introduce two new points whose positions are determined by the variables $t_0, t_m, \dots, t_n$, see Figure 4.5:

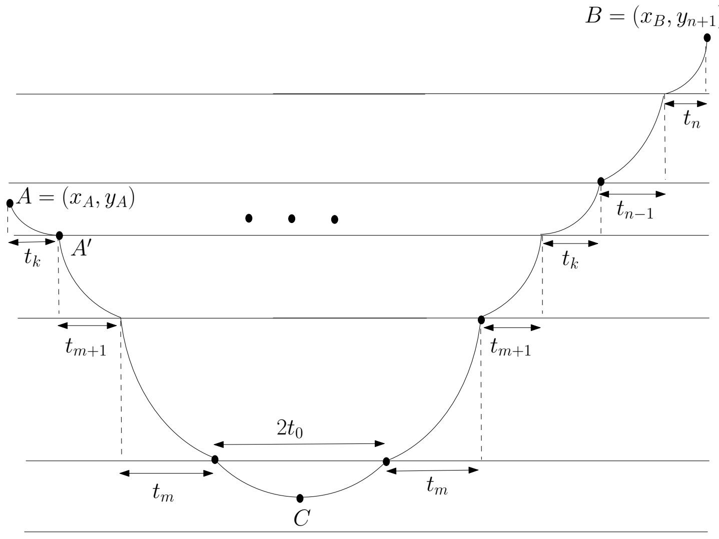


Figure 4.5: Arbitrary position of the starting point

- A' is the point where the line $y = y_k$ intersects the time-optimal trajectory for the first time, $A' = (x_B - \sum_{i=m}^{n} t_i - 2t_0 - \sum_{i=m}^{k} t_i, y_k)$,
- C is the lowest point of the time-optimal trajectory, $C = (x_B - \sum_{i=m}^{n} t_i - t_0, y_0^m - R_m(\bar{t}))$.

Now we consider the time-optimal trajectory from the point C to the point B. There are three types of equations that should be satisfied:

1) Equations (4.2) for $k = m+1, \ldots, n$, as this case is identical to the case of Section 4.2.

2) Equation $q_m R_m(\bar{t}) = q_{m+1} R_{m+1}(t_m)$ with the same reasoning.

$$\begin{aligned} R_m(\bar{t}) &= \sqrt{t_0^2 + (y_0^m - y_m)^2}, \text{ thus} \\ 2q_m t_m t_0 &= q_{m+1}(a_{m+1}^2 + t_m^2). \end{aligned}$$

3) $(x_0^{k+1}(\bar{t}) - x_A)^2 + (y_0^{k+1} - y_A)^2 = (x_0^{k+1}(\bar{t}) - x_{A'})^2 + (y_0^{k+1} - y_k)^2$, where $(x_0^{k+1}(\bar{t}), y_0^{k+1})$ is the center of the first circle segment within the layer L_{k+1}. The reason is that both A and A' belong to this circle segment. The x-coordinate of the center is

$$x_0^{k+1}(\bar{t}) = -\frac{y_k - y_A}{x_{A'} - x_A} y_0^{k+1} - \frac{x_A^2 + y_A^2 - x_{A'}^2 - y_k^2}{2(x_{A'} - x_A)}.$$

Denote $a^2 = (y_A - y_k)(2y_0^{k+1} - y_A - y_k)$ and $b^2 = (y_{k+1} - y_A)(2y_0^{k+1} - y_{k+1} - y_A)$, then the set of equations characterizing the arbitrary choice of the starting point is given in the following way:

$$
\begin{aligned}
q_{k+1}(a_{k+1}^2 - t_k^2)t_{k-1} &= q_k(a_k^2 + t_{k-1}^2)t_k, \; k = m+1, \ldots, n, && (4.9)\\
2q_m t_m t_0 &= q_{m+1}(a_{m+1}^2 + t_m^2), && (4.10)\\
t_k^2((x_{A'} - x_A)^2 + a^2)^2 &= (x_{A'} - x_A)^2((a_{k+1}^2 + t_k^2)^2 - 4b^2t_k^2), && (4.11)\\
t_i \in (0, a_{i+1}), i &\in \{m, \ldots, n\}, t_0 > 0, && (4.12)
\end{aligned}
$$

where $x_{A'} = x_B - \sum_{i=m}^{n} t_i - \sum_{i=m}^{k} t_i - 2t_0$.

Therefore, we can solve the general helicopter problem without obstacles for the velocity function with n breakpoints and arbitrary position of the start and end points.

4.5 Non-convexity of the time function

In this section we show that the time function $t(t_1, \ldots, t_n)$ is not convex, thus, the standard methods (for example, Newton's method) are not applicable directly. It is the case already for $n = 1$, for the velocity function with one breakpoint.

Consider the following function:

$$t(t_1) = \frac{1}{q_1}\int_0^{\gamma_1(t_1)} \frac{d\alpha}{\sin(\alpha + \beta_1(t_1))} + \frac{1}{q_2}\int_0^{\gamma_2(t_1)} \frac{d\alpha}{\sin(\alpha + \beta_2(t_1))}.$$

The first derivative is equal to (see Lemma 3.2.1):

$$\frac{dt(t_1)}{dt_1} = \frac{1}{q_2 R_2(t_1)} - \frac{1}{q_1 R_1(t_1)},$$

hence, the second derivative is calculated in the following way:

$$
\begin{aligned}
\frac{d^2t(t_1)}{dt_1^2} &= \frac{1}{q_2}\frac{-\frac{dR_2(t_1)}{dt_1}}{R_2^2(t_1)} - \frac{1}{q_1}\frac{-\frac{dR_1(t_1)}{dt_1}}{R_1^2(t_1)} = \\
&= \frac{q_2 R_2^2(t_1)\frac{a_1^4 - (x_B - x_A - t_1)^4}{4(x_B - x_A - t_1)^3 R_1(t_1)} + q_1 R_1^2(t_1)\frac{a_2^4 - t_1^4}{4t_1^3 R_2(t_1)}}{q_1 q_2 R_1^2(t_1) R_2^2(t_1)}.
\end{aligned}
$$

Let $x_A = 0$ and $x_B = 2(a_1 + a_2)$ then the second order derivative is negative for $t_1 = a_2$. The function $\frac{d^2t}{dt_1^2}(t_1)$ is continuous so there exists the interval $[a_2 - \varepsilon, a_2]$ such that the second order derivative is negative within this interval for sufficiently small ε. We conclude that the time function is non-convex.

4.5.1 Simple algorithm for the case of one breakpoint

We demonstrated above that the time function is not convex, thus in general it is impossible to apply Newton-type methods directly. Nevertheless, the case of one breakpoint admits a quick bijection algorithm, because the time function has one nice property; in particular, there is a unique stationary point which is a local minimum. In this subsection, first, we prove the existence, and second, we present the bijection algorithm.

Assume there exists a stationary point of the time function. Any stationary point should satisfy two following conditions:

- $q_1 R_1(t_1) = q_2 R_2(t_1)$ by Equation (4.1),
- $q_2(a_2^2 - t_1^2)(x_B - x_A - t_1) = q_1(a_1^2 + (x_B - x_A - t_1)^2)t_1$ by Equation (4.2), thus

$$q_1(a_1^2 + (x_B - x_A - t_1)^2) = \frac{q_2(a_2^2 - t_1^2)(x_B - x_A - t_1)}{t_1}.$$

We multiply both fractions in the numerator of the second order derivative $\frac{d^2 t(t_1)}{dt_1^2}$ by 1, but presented in the way $\frac{q_1^2}{q_1^2}$ and $\frac{q_2^2}{q_2^2}$ respectively, thus we can rewrite the expression using the first equation above in the following way:

$$\begin{aligned}\frac{d^2 t(t_1)}{dt_1^2} &= \frac{q_1 R_1(t_1)}{4(q_1 R_1(t_1))^4}\left[\frac{q_1^2(a_1^4 - (x_B - x_A - t_1)^4)}{(x_B - x_A - t_1)^3} + \frac{q_2^2(a_2^4 - t_1^4)}{t_1^3}\right] = \\ &= \frac{q_1^2(a_1^2 - (x_B - x_A - t_1)^2)(a_1^2 + (x_B - x_A - t_1)^2)}{4(q_1 R_1(t_1))^3 (x_B - x_A - t_1)^3} + \\ &+ \frac{1}{4(q_1 R_1(t_1))^3}\frac{q_2^2(a_2^2 - t_1^2)(a_2^2 + t_1^2)}{t_1^3}.\end{aligned}$$

Now we apply the second equation:

$$\begin{aligned}\frac{d^2 t(t_1)}{dt_1^2} &= \frac{q_1(a_1^2 - (x_B - x_A - t_1)^2) q_2(a_2^2 - t_1^2)(x_B - x_A - t_1)}{4(q_1 R_1(t_1))^3 t_1 (x_B - x_A - t_1)^3} + \\ &+ \frac{1}{4(q_1 R_1(t_1))^3}\frac{q_2^2(a_2^2 - t_1^2)(a_2^2 + t_1^2)}{t_1^3} = \\ &= \frac{q_2(a_2^2 - t_1^2)}{4(q_1 R_1(t_1))^3 t_1}\left[\frac{q_1(a_1^2 - (x_B - x_A - t_1)^2)}{(x_B - x_A - t_1)^2} + \frac{q_2(a_2^2 + t_1^2)}{t_1^2}\right] = \\ &= \frac{q_2(a_2^2 - t_1^2)}{4(q_1 R_1(t_1))^3 t_1}\left[\frac{q_1 a_1^2}{(x_B - x_A - t_1)^2} - q_1 + \frac{q_2 a_2^2}{t_1^2} + q_2\right].\end{aligned}$$

So, $\frac{d^2 t(t_1)}{dt_1^2} \geq 0$ as $t_1 < a_2$ and $q_2 > q_1$. We conclude that all stationary points are the points of local minimum in case of one breakpoint.

The time function is continuous, therefore if there is a stationary point then it is unique (and it is local minimum), as otherwise we will have a point of local maximum between two points of local minimum. Therefore, the time function monotonically decreases before this point and monotonically increases after this point. We can apply a simple bijection algorithm to find this point with any chosen level of precision. First we take the middle point x of the interval $[0, a_2]$ and two points in its $\frac{\varepsilon}{2}$-neighborhood. Depending on the values of the time function in these three points we choose the interval to the left or to the right of x to proceed; the second interval is excluded from the further consideration as it does not contain the point of local minimum. Therefore, in time O$(log(\frac{1}{\varepsilon}))$ we will find the value of x which is in ε-neighborhood of the local minimum.

4.6 Conclusions and open problems

In this chapter we have addressed the general problem of determining a time-optimal helicopter trajectory between two points in two-dimensional space, where the speed of the helicopter is a piece-wise linear concave function on the altitude. We used the motions primitives from the previous chapter to reduce the problem to the set of multivariate polynomial equations. If the number of velocity function breakpoints is constant then it enables us to solve the general helicopter problem in polynomial time using the algebraic elimination theory.

There are several interesting research directions that can be followed. First, it is an open question whether the general helicopter problem with polygonal obstacles in 2D admits a polynomial time algorithm. Second, are there efficient polynomial time approximation schemes for the general helicopter problem with obstacles in 3D? Notice, the three-dimensional problem with obstacles is NP-hard as it generalizes the Euclidean shortest-path problem with obstacles in 3D.

Part II

Exploiting Geometry in Graph Parameter's Determination

Chapter 5

Grid Minors

This chapter of the thesis is dedicated to treewidth in the class of planar graphs. More specifically, we examine the relation between treewidth and two other graph parameters, namely branchwidth and the side size of a largest square-grid minor in some special classes of planar graphs.

The complexity of finding the treewidth in planar graphs is not known, but it is widely believed that the problem is *NP*-hard. Approximating the treewidth of planar graphs is therefore an interesting research direction. To the best of our knowledge, the most successful approximations to the treewidth in planar graphs are based on two well known lower bounds, the branchwidth and the side size of the largest square-grid minor.

Branchwidth is a notion that is very closely related to treewidth and, just as for treewidth, it is an important algorithmic concept that is widely used in discrete mathematics and theoretical computer science; see, e.g., Bodlaender [16] and Hicks [59]. Gu and Tamaki [55] constructed a $\mathrm{O}(n^3)$ algorithm to compute optimal branch decompositions for planar graphs. For more work on planar branch decompositions, we refer to [12, 57, 58]. As for the relation between treewidth (tw) and branchwidth (bw), Robertson and Seymour show in [83] that for any graph G with $bw(G) > 1$ it holds that $bw(G) \leq tw(G) + 1 \leq \lfloor \frac{3}{2} bw(G) \rfloor$. Seymour and Thomas developed an $\mathrm{O}(n^3)$ time algorithm in [87] finding the minimum branchwidth of a given planar graph on n vertices. Therefore we can approximate treewidth efficiently in planar graphs to a factor $\frac{3}{2}$ from optimal.

Concerning the relation between treewidth and grid minors in planar graphs, independently in [52] and [89] it was shown that the treewidth of a planar graph is at most 5 times the side size of a largest square-grid minor in the graph. This upper bound on the treewidth was improved in [56] to 4.5 times the side size of a largest square-grid minor. The problem of finding the largest square-grid minor in a planar graph is interesting in itself. Although it is not known whether the problem can be solved in polynomial time, there is a $O(n^2 \log n)$ time algorithm for finding a square-grid minor with side size at least one fourth of the side size of the largest square-grid minor, see [18].

The definition of branchwidth (see Section 5.1) straightforwardly implies that the side size of the largest square-grid minor (gm) is bounded from above by the branchwidth in any graph. In [40], a short proof can be found for the fact that graphs with high branchwidth contain large grid minors. In this chapter we further analyze relationships between treewidth, branchwidth and the side size of the largest square-grid minor in planar graphs. We present a class of planar graphs for which $tw \approx \frac{3gm}{2} - 1$ and $bw = gm$. We experienced that this seemingly trivial task is quite challenging because there are no simple techniques to accurately estimate treewidth in planar graphs. For the presented graph family, the branchwidth is easily verifiable, while arguing why the treewidth in the presented graph class is large is quite technical and requires methods from graph minor theory.

We present two different ways to determine the parameter gm in our graph class. We see the contribution of this research therefore not only in the construction of lower bounds for the treewidth approximation, but also in the proof methodology for determining the parameter gm in a planar graph.

Furthermore we introduce two classes of planar graphs for which we conjecture that both branchwidth and treewidth are roughly equal to $2gm$. We do believe that these classes are worst cases for branchwidth and treewidth approximation in terms of gm in planar graphs.

The chapter is organized as follows. In Section 5.1, some new notions and definitions will be put forward. Subsequently in Section 5.2 we introduce a class of planar graphs for which we were able to determine bw, tw and gm. Two more classes of planar graphs are introduced in Section 5.3 for which several upper and lower bounds on bw, tw and gm are presented. Finally, we summarize the results from this chapter in Section 5.4 in which we also pose some open questions.

5.1 Preliminaries

For $n, m \geq 2$, the $(n \times m)$-*grid graph* (see [83]) is the simple graph with vertices v_{ij} $(1 \leq i \leq n,\ 1 \leq j \leq m)$ where v_{ij} and $v_{i'j'}$ are adjacent if $|i - i'| + |j - j'| = 1$. In this thesis, we are interested only in square grids. For simplicity of notation, we refer to the square $(n \times n)$-grid graph simply as the n-grid. In an n-grid, n is referred to as the side size of the n-grid.

We recall that H is a *minor* of a graph G if H is obtainable from a subgraph of G by edge contractions. A minor of graph G that forms a square grid graph is called a square-grid minor of G. The side size of the largest square-grid minor of G will be denoted by $gm(G)$.

We continue with definitions of branchwidth and branch decompositions. Both notions were introduced in [83].

Definition 5.1.1. *A* branch decomposition *of a graph $G = (V, E)$ is a pair (T, τ), where T is a ternary tree (every vertex has degree 1 or 3) and τ is a bijection from the set of leaves of T to $E(G)$.*

The *order* of an edge e of tree T in a branch decomposition is the number of vertices v from $V(G)$ such that there are leaves t_1, t_2 of T in different components of $T \setminus e$ for which $\tau(t_1), \tau(t_2)$ are both incident to v. The *width* of branch decomposition (T, τ) is the maximum order over all edges of T.

Definition 5.1.2. *The* branchwidth $bw(G)$ *of graph G is the minimum width over all branch decompositions of G (or 0 if $|E(G)| \leq 1$, when G has no branch decompositions).*

Two subsets $V', V'' \subseteq V$ in $G = (V, E)$ are said to *touch* each other if either they have a vertex in common or E contains an edge uv with $u \in V'$ and $v \in V''$. Given a graph $G = (V, E)$, we say that $V' \subseteq V$ is connected if $G[V']$ is connected, see Section 2.3.

Definition 5.1.3. *A set $\mathcal{B}$ of mutually touching, connected subsets of V is called a* bramble *of G. A subset of V intersecting with every element of $\mathcal{B}$ is called a* hitting set *for bramble $\mathcal{B}$.*

The *order* of a bramble $\mathcal{B}$ is the minimum size over all hitting sets for $\mathcal{B}$.

Definition 5.1.4. *The* bramble number *of a graph G is the maximum order over all brambles $\mathcal{B}$ of G.*

Brambles are a useful tool in bounding treewidth from below as can be concluded from the following theorem that is due to Seymour and Thomas [86] and its corollary.

Theorem 5.1.5. *Let k be a non-negative integer. A graph has treewidth k if and only if it has bramble number $k + 1$.*

Corollary 5.1.6. *Given graph G and a bramble $\mathcal{B}$ of order k, $tw(G) \geq k - 1$.*

Given a planar graph G together with its planar embedding, two faces f' and f'' in the embedding are said to be *adjacent* if there is a vertex in $V(G)$ incident to both f' and f''. We call a sequence $f_1, \ldots, f_n$ of faces an *$f_1 f_n$-facepath* if each of the faces in the sequence is adjacent to the previous face and to the next face in the sequence. The *length* of a facepath is equal to the number of faces in the facepath minus one.

Given an embedding of a graph G, we denote the collection of faces that are incident to vertex v by F_v. The *face distance* between faces f' and f'' in the embedding, denoted by $d_G(f', f'')$ in this thesis, is equal to the length of the shortest $f'f''$-facepath. The distance between vertex v and face f', denoted by $d_G(v, f')$, is equal to the length of a shortest $f'f''$-facepath, over all faces $f'' \in F_v$. It can be shown that for the considered planar graph families both distances are independent of the embedding of G, since they are almost 3-connected; see [23].

For two graphs G and H, the cartesian product $G \times H$ is a graph with

$$\begin{aligned} V(G \times H) &= V(G) \times V(H), \\ E(G \times H) &= \{\{(u, v_1), (u, v_2)\} \mid u \in V(G), \{v_1, v_2\} \in E(H)\} \bigcup \\ &\quad \bigcup \{\{(u_1, v), (u_2, v)\} \mid \{u_1, u_2\} \in E(G), v \in V(H)\}. \end{aligned}$$

5.2 X-grids

In this section we start by introducing a family X_i of planar graphs, which we will call X-grids. Graph X_n from this family can be constructed by taking an n-grid, two $(n \times \lceil n/2 \rceil)$-grids and two $(\lceil n/2 \rceil \times n)$-grids. The four rectangular grids are connected via their long sides to the four sides of the square grid, as is illustrated in Figure 5.1.

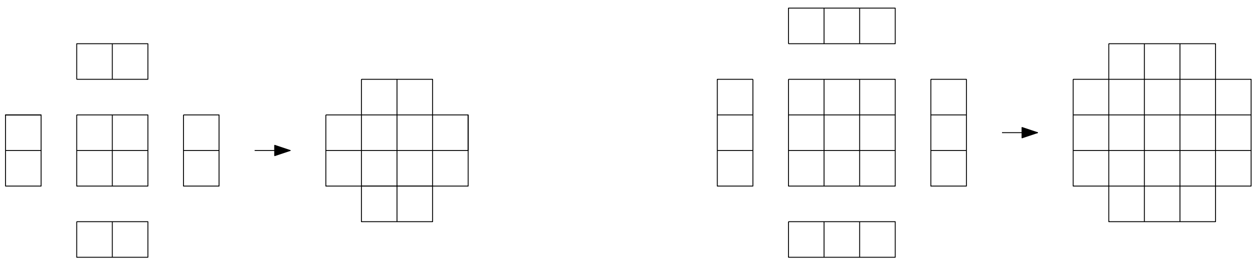


Figure 5.1: construction of X-grids X_3 and X_4

5.2.1 Branchwidth of X-grids

In this section, we study the branchwidth of X-grids. Our findings are encapsulated in the following theorem.

Theorem 5.2.1. *For the X-grid X_n, $bw(X_n) = n$.*

Proof. It is well known that the branchwidth of an n-grid is equal to n. Furthermore, for any minor H of graph G, $bw(H) \leq bw(G)$. Since the n-grid is a minor of X_n we conclude from the two foregoing observations that $bw(X_n) \geq n$.

To show that $bw(X_n) \leq n$, we construct a branch decomposition of X_n of width n. To do so, we split the edge set E of X_n up in four symmetrical parts $E_1, \ldots, E_4$, as is done in Figure 5.2 for X_4. Note that for both even and odd values of n, it is easy to make a partition of E in X_n consisting of 4 symmetrical parts.

Ternary tree T of the branch-decomposition consists of one middle edge and four symmetrical subtrees T_i that contain the leaves of T corresponding to edges from set E_i, for $i = 1 \ldots 4$. The middle edge of T has order equal to n, the edges of T that are incident to a leaf have order 2 and all other edges in T have order at most n. The width of the branch decomposition for X_n is therefore equal to n.

Figure 5.3 illustrates how to construct such an optimal branch decomposition of X_4. Each edge of X_4 in Figure 5.2 is denoted by a number. In Figure 5.3, the bijection τ between the leaves of T and E is displayed by simply putting these numbers in the leaves of T. The labels on edges e of T denote the set of vertices v of X_4 for which there are leaves t_1, t_2 of T in different components of $T \setminus e$, with $\tau(t_1), \tau(t_2)$ both incident to v. Following the same idea as is illustrated in Figures 5.2 and 5.3, it is easy to construct a branch decomposition of width n for X_n for any value of $n \geq 2$. □

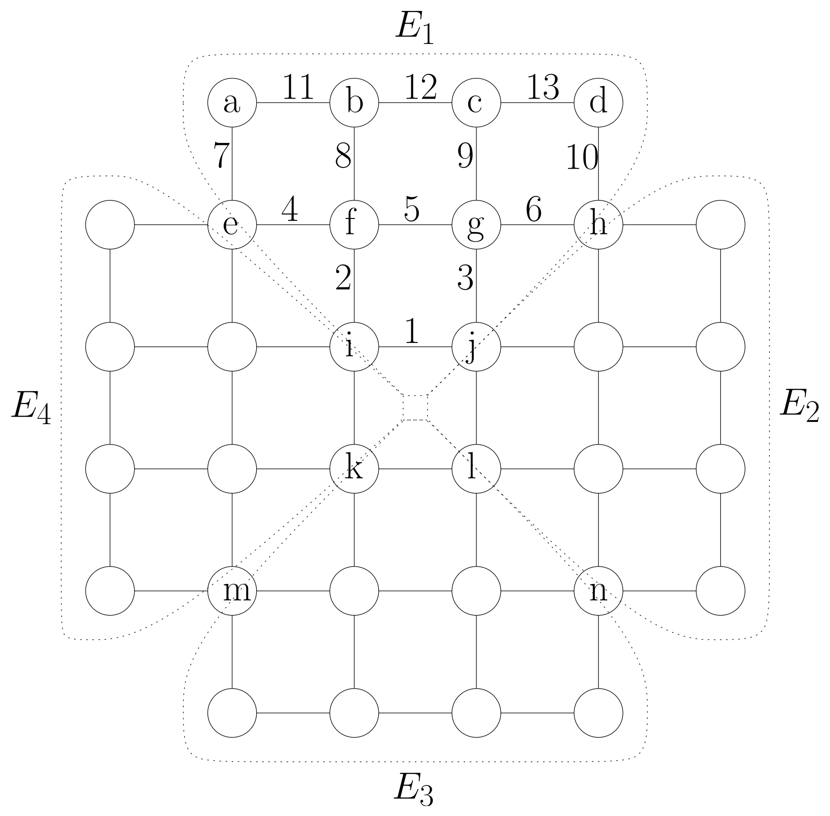


Figure 5.2: partition of E into sets $E_1, \ldots, E_4$ in planar graph X_4

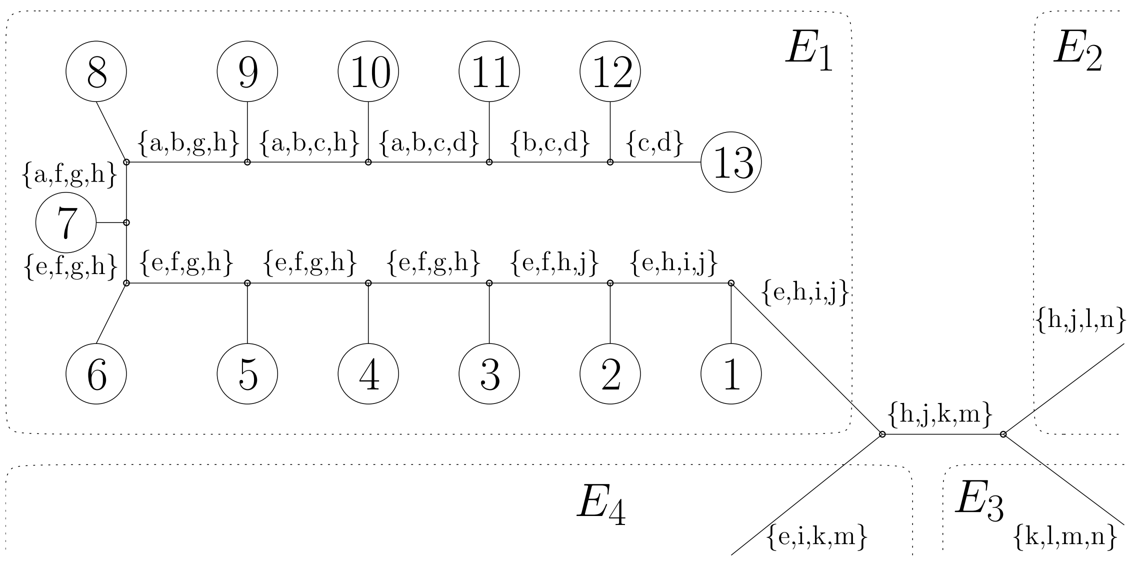


Figure 5.3: an optimal branch decomposition for the graph from Figure 5.2

5.2.2 Treewidth of X-grids

In this section, we take a closer look at the treewidth of X-grids. The results of this study is condensed in the following theorem.

Theorem 5.2.2. *For X-grid* X_n*,* $\lceil \frac{3n}{2} \rceil - 2 \leq tw(X_n) \leq \lfloor \frac{3n}{2} \rfloor - 1$.

Proof. It is easy to construct a tree decomposition for X_n of width $\lfloor \frac{3n}{2} \rfloor - 1$. Another way to prove that $tw(X_n) \leq \lfloor \frac{3n}{2} \rfloor - 1$ is to combine the result from Theorem 5.2.1 with the fact that $tw \leq \lfloor \frac{3bw}{2} \rfloor - 1$.

To prove that $tw(X_n) \geq \lceil \frac{3n}{2} \rceil - 2$, we construct a bramble $\mathcal{B}$ of X_n of order $\lceil \frac{3n}{2} \rceil - 1$. From this bramble it follows that the bramble number of X_n is at least $\lceil \frac{3n}{2} \rceil - 1$. Then, we straightforwardly apply Theorem 5.1.5.

For an illustration of the bramble construction, we refer to Figure 5.4(a-b). We split the rows of X_n into sets R_1, R_2 and R_3 and the columns into sets C_1, C_2 and C_3. The bramble $\mathcal{B}$ now consists of all subsets that are of one of the following four types.

1. The vertices from one row and from one column intersecting this row.
2. The vertices from one column from C_1, one column from C_3 and a path between these two columns;
3. The vertices from one row in R_3, one column in C_1 and a path between this row and column;
4. The vertices from one row in R_3, one column in C_3 and a path between this row and column.

One subset of $\mathcal{B}$ of type 1 and one subset of type 2 are depicted in Figure 5.4(a) with respectively fat solid lines and fat dashed lines. Subsets of $\mathcal{B}$ of type 3 and 4 are depicted respectively by the fat solid lines and the fat dashed lines in Figure 5.4(b). It can be easily verified that the subsets of $\mathcal{B}$ are connected and mutually touching.

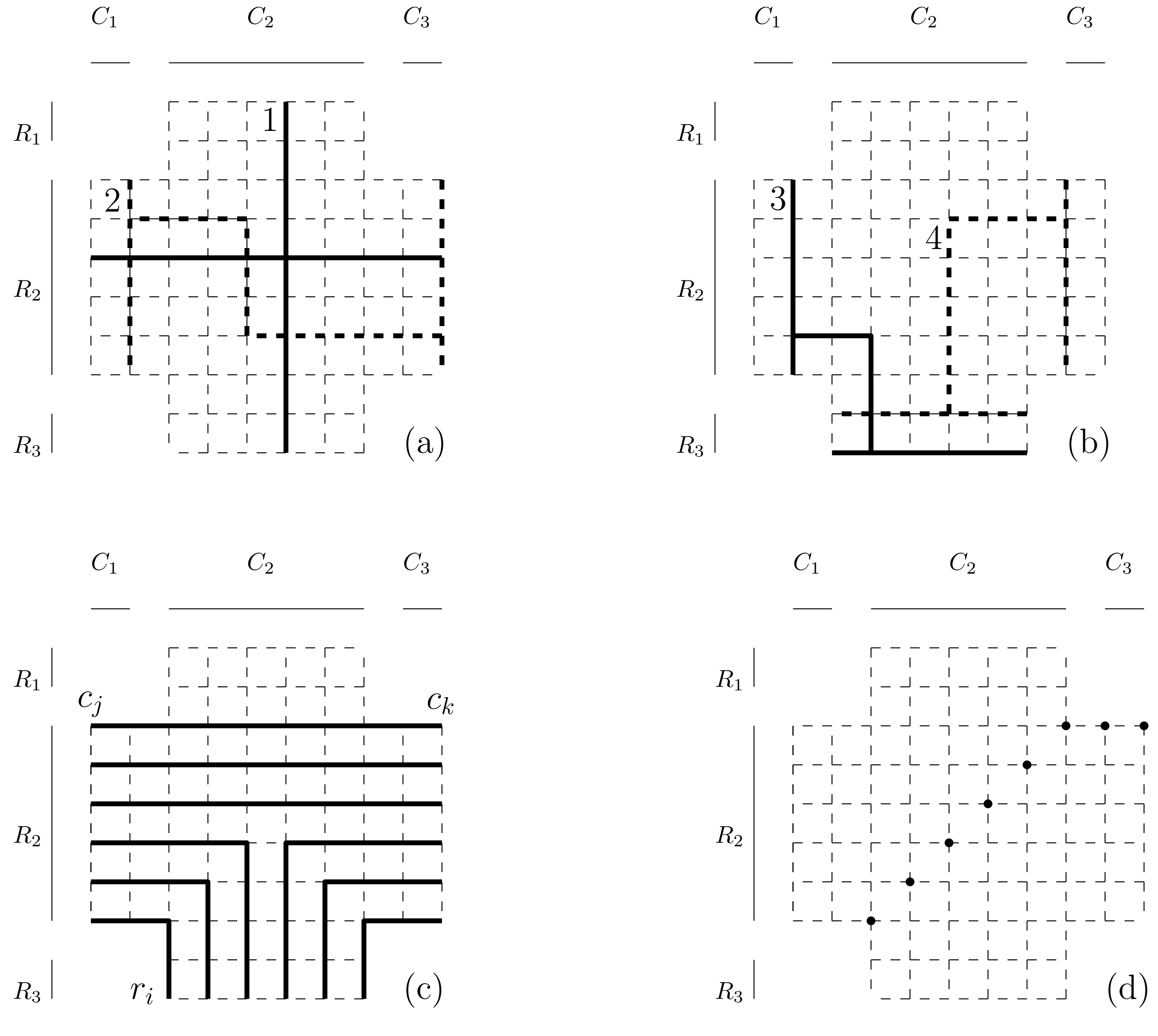


Figure 5.4: Examples of the four types of subsets of $\mathcal{B}$ in (a,b), a collection of $\lfloor \frac{3n}{2} \rfloor$ vertex disjoint paths in (c) and a hitting set for $\mathcal{B}$ of size $\lceil \frac{3n}{2} \rceil - 1$ in (d).

We will now show that the order of $\mathcal{B}$ is equal to $\lceil \frac{3n}{2} \rceil - 1$ which will thus imply that $tw(X_n) \geq \lceil \frac{3n}{2} \rceil - 2$.

First we prove that the order of $\mathcal{B}$ is at least $\lceil \frac{3n}{2} \rceil - 1$ by showing that for every vertex set V' such that $|V'| < \lceil \frac{3n}{2} \rceil - 1$, there is a subset of $\mathcal{B}$ that is not hit by V'. Consider such a set V'. Note that R_2 and R_3 together have $\lceil \frac{3n}{2} \rceil - 1$ rows and C_1, C_2 and C_3 have more than $\lceil \frac{3n}{2} \rceil - 1$ columns together. Thus, if every row in R_2 is hit by V', then there is a row in R_3 that is not hit by V'. Similarly, if every column in C_2 is hit by V', then there is a column in C_1 and a column in C_3 that are not hit by V'. Therefore, for any V' with $|V'| < \lceil \frac{3n}{2} \rceil - 1$, at least one of the following 4 situations occurs:

S1. Row r_i from R_2 and column c_j from C_2 are not hit by V'.

S2. Row r_i from R_2 and column c_j from C_1 are not hit by V'.

S3. Row r_i from R_3 and column c_j from C_2 are not hit by V'.

S4. Row r_i from R_3, column c_j from C_1 and column c_k from C_3 are not hit by V'.

We show that in each of these 4 situations, there is a subset of $\mathcal{B}$ that is not hit by V'. In situation S1, S2 and S3, r_i and c_j define a subset of $\mathcal{B}$ of type 1. Situation S4 needs somewhat more attention. First we point out that any vertex set in X_n that separates r_i from c_j and from c_k and simultaneously separates c_j from c_k contains at least $\lfloor \frac{3n}{2} \rfloor$ vertices. See Figure 5.4(c), where $\lfloor \frac{3n}{2} \rfloor$ vertex disjoint paths between r_i, c_j and c_k are depicted. Since V' contains strictly less than $\lceil \frac{3n}{2} \rceil - 1$ vertices and thus strictly less than $\lfloor \frac{3n}{2} \rfloor$ vertices, in situation S4 there exists thus a path P either between c_j and c_k or between r_i and c_j or between r_i and c_k such that P is not hit by V'. In the first case, $\{c_j, P, c_k\}$ forms a subset of $\mathcal{B}$ of type 2 that is not hit by V'. In the second case $\{r_i, P, c_j\}$ forms a subset of type 3 of $\mathcal{B}$ that is not hit by V' and in the last case $\{r_i, P, c_k\}$ forms a subset of $\mathcal{B}$ of type 4 that is not hit by V'. This shows that any set V' of size strictly smaller than $\lceil \frac{3n}{2} \rceil - 1$ can not be a hitting set of bramble $\mathcal{B}$. Hence the order of $\mathcal{B}$ is at least $\lceil \frac{3n}{2} \rceil - 1$.

To show that the order of $\mathcal{B}$ is equal to $\lceil \frac{3n}{2} \rceil - 1$, we note that the vertex set S as depicted in Figure 5.4(d) forms a hitting set of $\mathcal{B}$ of size $\lceil \frac{3n}{2} \rceil - 1$. It is easy to verify that all four types of subsets in $\mathcal{B}$ are indeed hit by this hitting set. □

5.2.3 Largest square-grid minor of X-grids

By construction, X_n contains an n-grid minor as an induced subgraph. Intuitively, it might seem clear that the side size of the largest square-grid minor in X_n is equal to n. Next, we present two proofs that support this intuition. The first proof is an easy one and is based on results concerning branch decompositions. Without the use of branch decompositions however, finding a proof turned out to be less trivial than we thought. In our second proof, we do not resort to branch decompositions. Instead, we use arguments related to face distances in the graph.

Theorem 5.2.3. *Given X-grid X_n, $gm(X_n) = n$.*

Proof. When we combine the result of Theorem 5.2.1 with the fact that for any graph G it holds that $gm(G) \leq bw(G)$, we find that $gm(X_n) \leq n$. By construction, X_n contains an n-grid as an induced subgraph, from which we conclude that $gm(X_n) \geq n$. The two foregoing observations yield that $gm(X_n) = n$. □

Surprisingly enough, without the use of branchwidth it is rather difficult to prove Theorem 5.2.3. In the alternative proof that will be presented next, we use arguments in terms of face distances as they are defined in Section 5.1. This proof will provide a methodological technique in contrast to the branchwidth based proof.

Before we start with the alternative proof, we need several propositions. The propositions concern a mapping of the face set of a planar embedding of a graph G to the face set of a planar embedding of a minor M of G. We recall that a minor M can be obtained from G by a series of vertex deletions, edge deletions and edge contractions. Basically, if such operation does not change the number of faces in the embedding, then a face is mapped to the face in the embedding of the minor that naturally corresponds to it. If, by a vertex deletion or edge deletion, some faces are joined together to one face, then all these faces are mapped to the joined face. If, by an edge contraction, some face f disappears, then f is mapped to a face in the embedding of the minor that naturally corresponds to a neighbor face of f. We say that face f' in an embedding of minor M of G corresponds to face f in an embedding of G if f can be mapped to f'. We now introduce three propositions, for which correctness can be easily verified.

Proposition 5.2.4. *Using the mapping described above, each face in an embedding of G can be mapped to a face in an embedding of a minor M of G. Moreover, for any number i of different faces in the embedding of M, there are at least i different faces in the embedding of G that can respectively be mapped to them.*

Proposition 5.2.5. *Let f' and f'' be two faces in the embedding of G and let f'_M and f''_M be respectively the two faces corresponding to f' and f'' in the embedding of a minor M of G. Then, $d_M(f'_M, f''_M) \leq d_G(f', f'')$.*

Proposition 5.2.6. *Let f' be a face and v be a vertex in G and let M be a minor of G in which v is not deleted nor contracted to another vertex. Furthermore, let f'_M be a face in M corresponding to f'. Then, $d_M(v, f'_M) \leq d_G(v, f')$.*

Using these propositions, we are now ready to present an alternative proof of Theorem 5.2.3.

Proof. Again by construction of X_n, it is easy to see that X_n contains an n-grid as a minor. To show that $gm(X_n) = n$, we show that X_n has no $(n+1)$-grid minor. We will show that the outer face in an embedding of X_n cannot be mapped to any face in an embedding of the $(n+1)$-grid. Then by Proposition 5.2.4 we derive that the $(n+1)$-grid cannot be a minor of X_n. The proof is divided into three cases. For all three cases, we present arguments only for even values of n. For odd values of n the proof is similar. We consider the natural embedding of X_n and of the $(n+1)$-grid and from here on, we just talk about a face of X_n ($(n+1)$-grid) instead of about a face in the natural embedding of X_n ($(n+1)$-grid).

Case 1. The outer face of X_n cannot be mapped to the outer face of the $(n+1)$-grid. Because n is even, there is a vertex v in the $(n+1)$-grid that has distance $\frac{n}{2}$ to the outer face f', see the leftmost picture in Figure 5.5. If the outer face of X_n could be mapped to the outer face of the $(n+1)$-grid, then by Proposition 5.2.6 there should also be a vertex in X_n with distance at least $\frac{n}{2}$ to the outer face of X_n. However, there are no such vertices in X_n, see the rightmost picture in Figure 5.5.

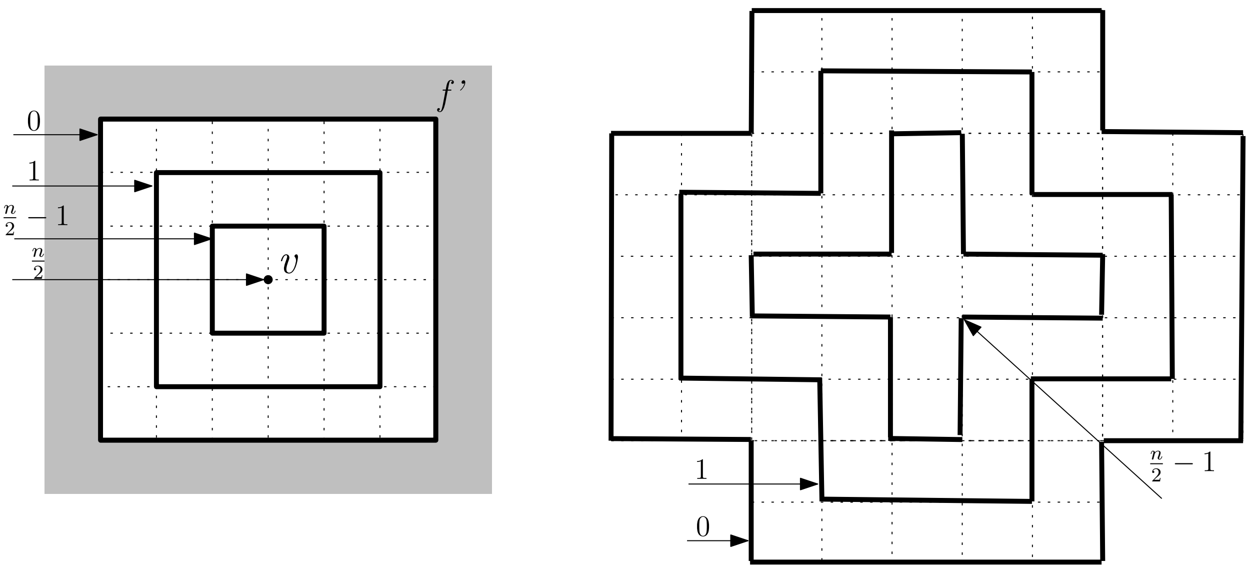
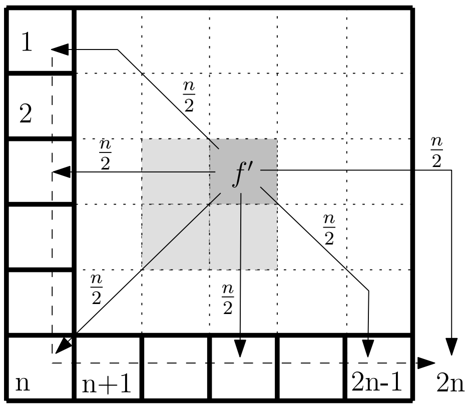


Figure 5.5: Distances of vertices to the outer face in the $(n+1)$-grid and in X_n

Case 2. We now show that the outer face of X_n cannot be mapped to any of the 4 middle faces of the $(n+1)$-grid. There are $2n$ faces in the $(n+1)$-grid having face distance $\frac{n}{2}$ to a middle face f', see Figure 5.6, leftmost picture. Suppose that the outer face of X_n can be mapped to any of the 4 middle faces of the $(n+1)$-grid. By Propositions 5.2.4 and 5.2.5 then there must be $2n$ different faces in X_n at face distance at least $\frac{n}{2}$ to the outer face. However, there are only $2n-3$ such faces in X_n, see the rightmost picture in Figure 5.6.

Case 3. Finally, let us show that the outer face of X_n cannot be mapped to any of the other inner faces of the $(n+1)$-grid. For each such inner face of the $(n+1)$-grid there is a vertex at distance $\frac{n}{2}$, see Figure 5.7. Suppose that the outer face of X_n can be mapped to one of the other inner faces of the $(n+1)$-grid. Then, by Proposition 5.2.6



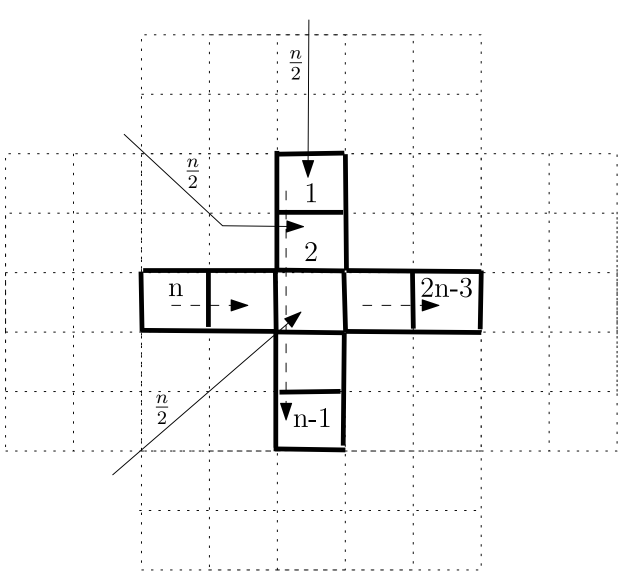


Figure 5.6: There are $2n$ faces at face distance $\frac{n}{2}$ to f' in the $(n+1)$-grid and $2n-3$ faces at face distance $\frac{n}{2}$ to the outer face in X_n

there must be a vertex in X_n that has distance at least $\frac{n}{2}$ to the outer face. Again, there is no such vertex in X_n, see the rightmost picture in Figure 5.5. □

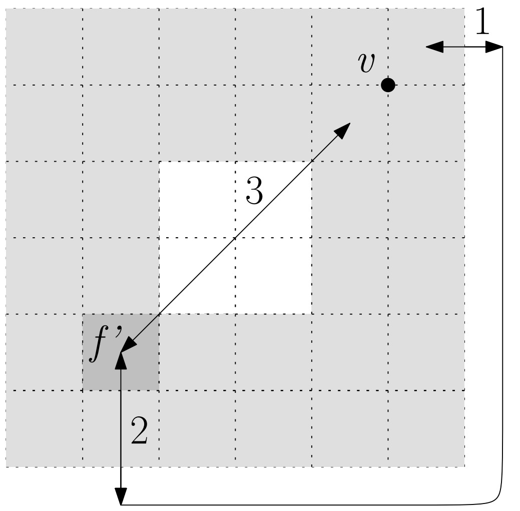


Figure 5.7: Vertex v at distance $\frac{n}{2}$ to a non-middle inner face f' in the $(n+1)$-grid

5.3 Sandwich grids and pyramids

In this section, we introduce two more classes of planar graphs, which we call *pyramids* and *sandwich grids*. The graphs in the pyramid family will be referred to as

Λ_n, whereas S_n will be used to denote a sandwich grid. The pyramid Λ_n is a graph on $2n^2 - 2n + 1$ vertices and $6n^2 - 10n + 4$ edges. It can be constructed by building a pyramid with the side size of the base level equal to n, as shown in the leftmost picture in Figure 5.8. The sandwich-grid S_n is a graph on $2n^2$ vertices and $4n^2 - 4$ edges and can be constructed by taking two n-grids and connecting the vertices on the outer face of one n-grid to the corresponding vertices on the outer face of the second n-grid, as is shown in the rightmost picture in Figure 5.8. Planar embeddings of the graphs Λ_3 and S_4 are given in Figure 5.9.

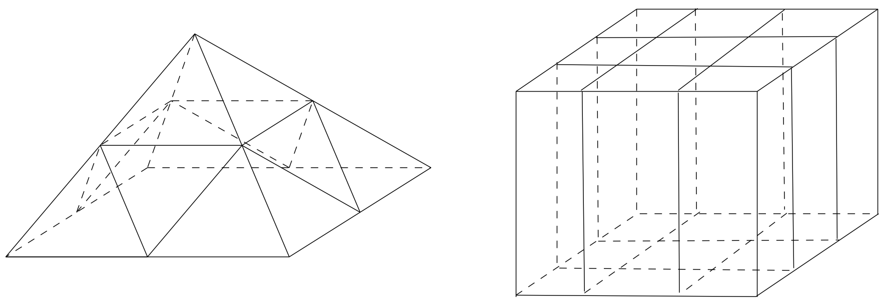


Figure 5.8: pyramid Λ_3 and sandwich grid S_4

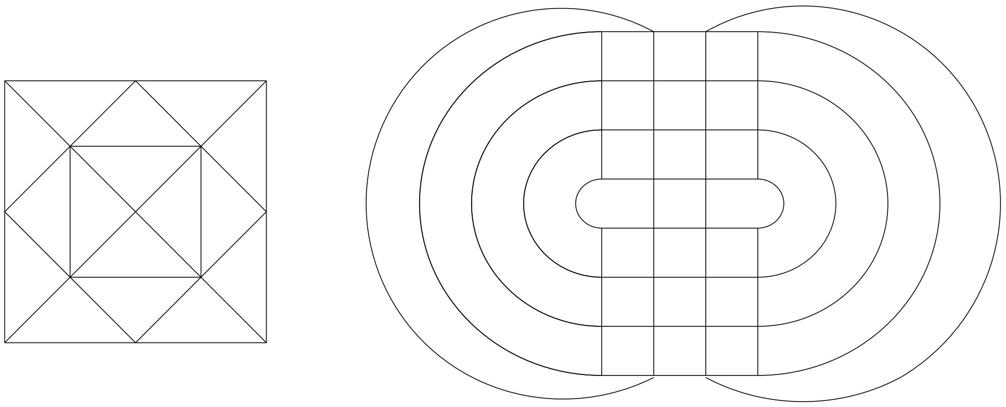


Figure 5.9: planar embeddings for pyramid Λ_3 and sandwich grid S_4

In the following we will determine the treewidth and branchwidth of sandwich grids. We will also present a lower bound on the side size of the largest square-grid minor for both families and an upper bound on the treewidth of pyramids. We conjecture that these bounds are tight.

5.3.1 Branchwidth of sandwich grids

The branchwidth of sandwich grids can be determined, but indirectly, using the result by Gu and Tamaki [56] which followed the publication based on this chapter. They introduced the family of graphs called *cylinders*.

Definition 5.3.1. *(see [56]) Let $k \geq 3$ and $h \geq 1$ be integers. A $k \times h$* cylinder*, denoted by $C_{k,h}$, is the Cartesian product of a cycle on k vertices and a path on h vertices: it has a vertex (u, v) for each vertex u of the cycle and each vertex v of the path and (u, v) is adjacent with (u', v') if and only if $u = u'$ and v is adjacent with v' on the path or $v = v'$ and u is adjacent with u' on the cycle.*

Gu and Tamaki proved that $bw(C_{2n,n}) = 2n$. In fact, cylinder $C_{2n,n}$ is a subgraph of sandwich grid S_n. Thus, we can use $bw(C_{2n,n})$ as a lower bound of $bw(S_n)$.

Theorem 5.3.2. *For a sandwich grid S_n, $2n \leq bw(S_n) \leq 2n + 1$.*

Proof. To show that $bw(S_n) \leq 2n + 1$, we use the result from the next section that $tw(S_n) \leq 2n$. When we combine this with the fact that for any graph G it holds that $bw(G) \leq tw(G) + 1$, we obtain the desired result.

To show that $bw(S_n) \geq 2n$, we use a result on cylinder graphs. Since the branchwidth of a subgraph of S_n is smaller than or equal to $bw(S_n)$, we conclude that $bw(S_n) \geq bw(C_{2n,n}) = 2n$. □

The question to determine the branchwidth of the pyramid class Λ_n still remains open. Later we will show that $tw(\Lambda_n) \leq 2n-1$. Using this result, we can bound $bw(\Lambda_n)$ from above by $2n$.

5.3.2 Treewidth of sandwich grids and pyramids

The treewidth of sandwich grids can be approximated as follows.

Theorem 5.3.3. *For sandwich grid S_n, $2n - 1 \leq tw(S_n) \leq 2n$.*

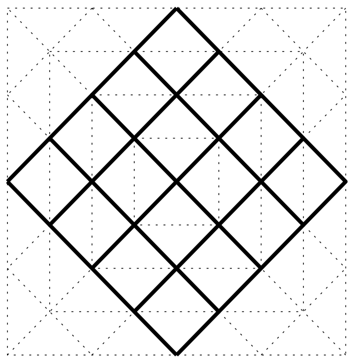
Proof. To prove that $tw(S_n) \leq 2n$, we show how to construct a tree decomposition (actually, a path decomposition) of S_n of width $2n$. One can start with a bag that contains $2n$ vertices from the first column of S_n (using the embedding of Figure 5.9) plus the top vertex of the second column. In the next bag, we eliminate the top vertex from the first column and introduce the second vertex from the second column, etc. The last bag contains the bottom vertex of the one-to-last column and the $2n$ vertices from the last column in S_n.

To show that $tw(S_n) \geq 2n - 1$, we use a result from [56]. A result from this study is that for subgraph $C_{2n,n}$ of S_n it holds that $bw(C_{2n,n}) = 2n$. Therefore $bw(S_n) \geq 2n$. Combined with the fact that $bw(S_n) \leq tw(S_n) + 1$ this implies that $tw(S_n) \geq 2n - 1$. □

For the pyramid Λ_n, we construct a tree decomposition showing that $tw(\Lambda_n) \leq 2n-1$. Consider one of the main diagonals in Λ_n (using the embedding from Figure 5.9) and all other paths in Λ_n that are parallel to this diagonal. Let the middle bag of the tree decomposition contain all vertices from the main diagonal. In both directions we add a bag containing the vertices from the main diagonal plus the top vertex from the next path. After that, we simply eliminate the vertices one by one. This will give a path decomposition of Λ_n of width $2n-1$, hence $tw(\Lambda_n) \leq 2n-1$. The question to prove tightness of this upper bound is still open at the moment of writing this thesis.

5.3.3 Square-grid minor of sandwich grids and pyramids

It is easy to see that $gm(S_n) \geq n$ since S_n contains the n-grid as an induced subgraph. To show that $gm(\Lambda_n) \geq n$, we refer to Figure 5.10, which illustrates how to obtain an n-grid minor in Λ_n for odd and even values of n. We do believe that both families S_n and Λ_n do not contain $(n+1)$-grid minors, but we are still looking for techniques to prove this.



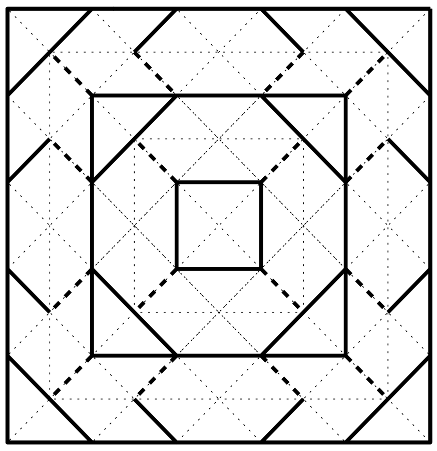


Figure 5.10: To obtain n-grid minors in Λ_n, delete dotted edges and vertices that are incident only to dotted edges and contract the dashed edges.

5.4 Conclusions and open questions

The results from this study are summarized in Table 5.1.

Furthermore, we compute the parameters for graphs X_n for small values of n, see Table 5.2. These computations provide some evidence that $\omega(X_n) = \lfloor \frac{3n}{2} \rfloor - 1$ for even

graph family	**gm**	**bw**	**tw**
X-grid (X_n)	n	n	$\approx \frac{3n}{2} - 1$
pyramid (Λ_n)	$\geq n$ $\leq 2n$	$\geq n$ $\leq 2n$	$\geq n$ $\leq 2n - 1$
sandwich grid (S_n)	$\geq n$ $\leq 2n$	$\approx 2n$	$\approx 2n - 1$

Table 5.1: summary of results

values of n.

Table 5.2: Values of the parameters of X_n for small n

Graph	$\theta(G)$	$\beta(G)$	$\omega(G)$
X_2	2	2	2
X_3	3	3	3
X_4	4	4	5
X_5	5	5	6

As a direction for further research, we recommend the interested reader to consider the following questions.

1. Can we find a technique to prove that $gm(\Lambda_n), gm(S_n) \leq n$?
2. Can we find a bramble for Λ_n of order $2n$, i.e., can we show that $tw(\Lambda_n) = 2n - 1$?
3. Can we find a family of planar graphs satisfying $c_1 gm < bw < c_2 tw$, where $c_1 > 1$ and $c_2 < 1$ are some constants? Notice, that in X_n there is a constant-ratio gap between $gm(X_n)$ and $bw(X_n)$, while in $C_{2n,n}$ there is a gap between $gm(C_{2n,n})$ and $tw(C_{2n,n})$.

We end this chapter with the conjecture that the sandwich grid, the pyramid and the cylinder(see [56]) are worst cases for branchwidth and treewidth approximation in terms of the side size of the largest square-grid minor.

Conjecture 5.4.1. *For any planar graph G, both $bw(G)$ and $tw(G)$ are at most $2gm(G)+ o(gm(G))$.*

Chapter 6

Integer Linear Programming Formulations for Treewidth

There are various algorithms and heuristics for the problem of finding the treewidth of a graph. One of the most known exact algorithms is Bodlaender's algorithm running in $O(2^{tw^2}n)$-time, where tw is the treewidth of a graph, see [14]. Unfortunately it can not be used in practice because of the big constant hidden in $O()$-notation. The best known exponential-time algorithm is the dynamic programm by Fomin et al. [47] with the running time $O(1.8899^n)$.

For a long time the best known approximation algorithm for the treewidth had a factor $O(\log n)$, see [17], and $O(\log tw)$, see [3]. Recently, Feige et al. developed a polynomial-time $O(\sqrt{\log tw})$-approximation algorithm, see [43]. It is still an intriguing open question whether the treewidth can be approximated within a constant factor for an arbitrary graph.

So far only limited attempts were made to attack the treewidth problem using Integer Linear Programming techniques. In this chapter we present an overview of these attempts and develop further LP-based techniques for the problem. Integer Linear programming is widely used to obtain good exact and approximation algorithms for different problems, see i.e. [24, 44, 72, 75, 78]. The advantage of this approach is that it is very generic and, therefore, we can apply many different well-developed LP-techniques to obtain better quality solutions or to speed-up the algorithm.

We present two different ILP formulations for the treewidth. The first ILP, the *Elimination Order formulation*, is proposed by Koster and Bodlaender [65]. It is based on the vertex elimination order and the connection between the treewidth and the chordalization of the graph, see Chapter 2, Theorem 2.6.8 for the details. In this thesis we briefly show that the integrality gap of this formulation is at least $\Omega(\sqrt{n})$. We improve this formulation by merging it with the flow metric approach by Bornstein and Vempala [24]. The second ILP, the *Tree Drawing formulation*, is the outcome of this thesis. This formulation has nice geometric properties; we use them to apply a local branching technique proposed by Fischetti and Lodi [46].

The chapter is organized as follows: Section 6.1 is devoted to the Elimination Order formulations and its improvement using flow metric techniques by Bornstein and Vempala. Section 6.2 contains new Tree Drawing formulation and the application of the local branching technique by Fischetti and Lodi to the proposed ILP. In Section 6.3 we compare two formulations and speculate on their applicability for different types of graphs. The last section contains the results and open questions.

6.1 Elimination Order formulation and flow metrics

The Elimination Order formulation for the treewidth is introduced by Bodlaender and Koster [65]. The formulation is based on the relationship between treewidth and chordalizations of graphs, see Chapter 2 for the details. Theorem 2.6.8 states that finding the treewidth of a graph G is equivalent to finding a triangulation of the graph G with minimum clique size. A graph is triangulated if and only if it has a *perfect elimination scheme*. The idea is to determine the best elimination order of the vertices. The maximum outdegree among the vertices in an elimination scheme is equal to the width of the tree decomposition of G associated with the triangulation. Modeling perfect elimination orders let the decision variables of ILP be defined as follows:

$$x_{ij} = \begin{cases} 1, & \text{if } i \text{ is ordered before } j \text{ in the perfect elimination scheme,} \\ 0, & \text{otherwise.} \end{cases}$$

$$y_{ij} = \begin{cases} 1, & \text{if } i \text{ is ordered before } j \text{ in the perfect elimination scheme} \\ & \quad \text{and if } ij \text{ is an edge of the triangulation,} \\ 0, & \text{otherwise.} \end{cases}$$

We assume that $i \neq j$ for all x_{ij} and y_{ij}.

Now, the elimination order based formulation (EOF) reads (see [65]):

$$\min w \tag{6.1}$$

subject to

$$w \geq \sum_{j \in V} y_{ij}, \qquad \forall i \in V, \tag{6.2}$$

$$x_{ij} + x_{ji} = 1, \qquad \forall \{i,j\} \subseteq V, \tag{6.3}$$

$$x_{ij} + x_{jk} - x_{ik} \leq 1, \qquad \forall \{i,j,k\} \subseteq V, \tag{6.4}$$

$$y_{ij} \leq x_{ij}, \qquad \forall \{i,j\} \subseteq V, \tag{6.5}$$

$$y_{ij} = x_{ij}, \qquad \forall ij \in E, \tag{6.6}$$

$$x_{jk} + y_{ij} + y_{ik} - y_{jk} \leq 2, \qquad \forall \{i,j,k\} \subseteq V, \tag{6.7}$$

$$x_{ij} \in \{0,1\}, y_{ij} \in \{0,1\}, \qquad \forall i,j \in V. \tag{6.8}$$

Constraints (6.3), (6.4) and (6.8) guarantee that variables x determine a linear order. Constraints (6.5) force the y_{ij} variable to be zero as soon as i is not ordered before j. Constraints (6.6) impose that edges of G are counted in the triangulation. Constraints (6.7) imply that if edges ij and ik are in the triangulation with i ordered before j and i ordered before k, then there must exist an edge jk in the triangulation.

First, let us estimate the integrality gap of EOF. Consider $m \times m$-grid graph which is a Cartesian product of two simple paths, $P_m \times P_m$ (Cartesian product is the direct product of two sets, see, e.g. [92]). The number of vertices in this graph is $n = m^2$ and $tw = m = \sqrt{n}$. It is easy to check that the symmetric solution

$$x_{ij} = \frac{1}{2}, \forall \{i,j\} \subseteq V,$$

$$y_{ij} = \begin{cases} \frac{1}{2}, & \text{if } ij \in E, \\ 0, & \text{otherwise.} \end{cases}$$

is feasible for the LP-relaxation where Constraints (6.8) are substituted just by non-negativity requirement. Given this solution w is maximal on the internal vertices of the grid which have degree 4, thus $LP \leq 4 * \frac{1}{2} = 2$ where LP is the optimal solution of the linear relaxation of EOF. Let OPT denotes the treewidth of the given graph. Then, the integrality gap of EOF is at least $\frac{OPT}{LP} \geq \frac{m}{2} = \frac{\sqrt{n}}{2}$.

6.1.1 Introduction to flow metrics

In this subsection we give a brief introduction to the flow metric technique introduced by Bornstein and Vempala [24] where the notion *flow metric* refers to the relaxation of the path metric (i.e. linear ordering). A lot of problems in combinatorial optimization can be formulated as finding an assignment of pairwise distances of a specified type which minimizes a special cost function on the distances. The problems modeled in this way are often NP-hard and one approach to solve them is to consider relaxations of the associated metric. The proposed method uses the same relaxation and essentially the same rounding procedure to solve the problem; only the objective function is varied.

Consider a path on n vertices numbered $1, 2, \dots, n$. The distance between u and v in the corresponding path metric is $|u - v|$.

Now imagine that we send a flow of one unit from a vertex u to a vertex v to the right of u for all pairs of vertices u and v such that $v > u$. These flows satisfy the following properties:

- For each pair of vertices u and v, the value of the flow from u to v plus the value of the flow from v to u is equal to one.
- For every triplet u, v, w, the flow between u and v that goes through w, the flow between u and w that goes through v and the flow between v and w that goes through u sum to one.

Any metric that satisfies the above properties is called a flow metric. To define this formally, let the variables $f_{ij}^{u,v}$ represent the value of the flow from u to v on the edge ij. These variables satisfy flow conservation at every vertex except the source and the sink. We pay attention that the variables $f_{ij}^{u,v}$ are determined only for edges $ij \in E$. Define a set of auxiliary variables

$$g_w^{u,v} = \sum_{i:\, iw \in E} f_{iw}^{u,v}.$$

In other words, $g_w^{u,v}$ represents the flow from u to v that goes through the vertex w.

Definition 6.1.1. *[24] A* flow metric *is a solution to the following linear program (FP), where f and g are flow variables as defined above.*

$$g_j^{i,j} + g_i^{j,i} = 1, \qquad \forall \{i,j\} \subseteq V, \tag{6.9}$$

$$(g_k^{i,j} + g_k^{j,i}) + (g_i^{j,k} + g_i^{k,j}) + (g_j^{i,k} + g_j^{k,i}) = 1, \qquad \forall \{i,j,k\} \subseteq V, \tag{6.10}$$

$$d_f(i,j) = \sum_{k \in V} g_k^{i,j}, \qquad \forall i,j \in V, \tag{6.11}$$

$$d_f(i,j) + d_f(j,k) \geq d_f(i,k), \qquad \forall i,j,k \in V. \tag{6.12}$$

Constraints (6.11) define a set of directed distances that form the "metric" and Constraints (6.12) impose the triangle inequality on these distances.

An important property of a flow metric is that it satisfies the one-dimensional spreading constraints as stated in the following theorem:

Theorem 6.1.2. *[24] For any subset $S \subseteq V$ of vertices,*

$$\sum_{u,v \in S} d_f(u,v) \geq \binom{|S|}{3}.$$

Roughly speaking, for a spreading metric the average distance in any subset of k vertices is about k, as in the case of a path. This property can be modeled using a set of linear constraints above. Although the number of constraints is exponential in the size of the input graph, they admit an efficient separation oracle [54] and so the resulting relaxations can be solved in polynomial time.

6.1.2 Improvement of EOF using flow metrics

Flow metric technique is a powerful tool of polishing the ordering polytope in the neighborhood of the optimal solution. We use its power to improve the Elimination Order formulation for treewidth. The way to merge ILP and flow metric technique is based on the comparison of the variables and identification of some of them. Consider two relaxations of linear ordering polytopes (6.3)–(6.4) and (6.9)–(6.10) we can straightforwardly combine these constraints into one single polytope in the following way:

Proposition 6.1.3.

$$x_{ij} = g_j^{i,j}.$$

Proof. We remind that x_{ij} and $g_j^{i,j}$ are defined as follows:

$$x_{ij} = \begin{cases} 1, & \text{if } i \text{ is ordered before } j \text{ in the perfect elimination scheme,} \\ 0, & \text{otherwise.} \end{cases}$$

$$g_j^{i,j} = \sum_{k\colon kj \in E} f_{kj}^{i,j} = \begin{cases} 1, & \text{if there is a flow from } i \text{ to } j, \\ 0, & \text{otherwise.} \end{cases} =$$

$$= \begin{cases} 1, & \text{if } i \text{ is before } j \text{ in the vertex linear ordering,} \\ 0, & \text{otherwise.} \end{cases}$$

We conclude that x_{ij} and $g_j^{i,j}$ are equal in the integral solution. □

In fact, by Proposition 6.1.3 we notice that Equation (6.3) and Equation (6.9) are identical. We formulate the merged ILP as follows:

$$\min w \tag{6.13}$$

subject to

$$w \geq \sum_{j \in V} y_{ij}, \qquad \forall i \in V, \tag{6.14}$$

$$g_k^{i,j} = \sum_{m \in V\colon mk \in E} f_{mk}^{i,j}, \qquad \forall\, i,j,k \in V, \tag{6.15}$$

$$g_j^{i,j} + g_i^{j,i} = 1, \qquad \forall \{i,j\} \subseteq V, \tag{6.16}$$

$$g_j^{i,j} + g_k^{j,k} - g_k^{i,k} \leq 1, \qquad \forall \{i,j,k\} \subseteq V, \tag{6.17}$$

$$y_{ij} \leq g_j^{i,j}, \qquad \forall \{i,j\} \subseteq V, \tag{6.18}$$

$$y_{ij} = g_j^{i,j}, \qquad \forall ij \in E, \tag{6.19}$$

$$g_k^{j,k} + y_{ij} + y_{ik} - y_{jk} \leq 2, \qquad \forall \{i,j,k\} \subseteq V, \tag{6.20}$$

$$(g_k^{i,j} + g_k^{j,i}) + (g_j^{i,k} + g_j^{k,i}) + (g_i^{j,k} + g_i^{k,j}) = 1, \qquad \forall \{i,j,k\} \subseteq V, \tag{6.21}$$

$$d_f(i,j) = \sum_{k \in V} g_k^{i,j}, \qquad \forall i,j \in V, \tag{6.22}$$

$$d_f(i,j) + d_f(j,k) \geq d_f(i,k), \qquad \forall i,j,k \in V, \tag{6.23}$$

$$g_k^{i,j} \in \{0,1\}, y_{ij} \in \{0,1\}. \tag{6.24}$$

We make the following conjecture about the presented ILP:

Conjecture 6.1.4. *Applying rounding algorithm by Bornstein and Vempala [24] to the linear relaxation of ILP (6.13)–(6.24), one derives an $O(\log n)$-approximate solution to the treewidth problem in polynomial time.*

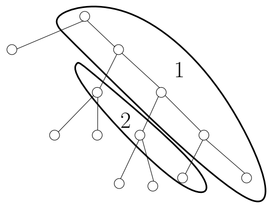


Figure 6.1: The construction is symmetric with respect to the root. We choose the longest path from the root to the bottom and merge it into one bag 1 of the path-decomposition. Bag 2 is connected to bag 1, and so on.

To show that the proposed method is promising and the conjecture is plausible, we present a following $O(\log^3 n)$-approximation for the treewidth. Consider the pathwidth problem, where condition on the tree T in Definition 2.6.1 is strengthened to the condition that T is a simple path. The pathwidth problem is formulated as an ILP on the polytope contained in (6.14)–(6.24), see Bornstein and Vempala [24]. They proposed the rounding algorithm which provides a solution to the pathwidth of value

$APX_{pw} \leq pw \times O(\log^2 n)$. Clearly, $tw \leq pw$. It is well known that $pw \leq tw \times O(\log n)$. The argument is as follows: without loss of generality we consider the binary tree-decomposition on n vertices. The depth of this tree is at most $\mathrm{O}(\log n)$. The way to pack the tree-decomposition into the path-decomposition is demonstrated on Figure 6.1. Every new bag of the path-decomposition is at most $tw \times O(\log n)$.

Therefore, if we simply take the derived value of pathwidth as a feasible approximate solution to the treewidth we have than

$$tw \leq pw \leq APX_{pw} = APX_{tw} \leq pw \times O(\log^2 n) \leq tw \times O(\log^3 n).$$

Thus, already the direct usage of the rounding algorithm for the pathwidth guarantees the polylogarithmic approximation for the treewidth. Conjecture 6.1.4 stays open as a question for further research. Notice, this polylogarithmic approximation immediately implies that we improved upon (6.1)–(6.8) by adding the flow metric constraints. This is because the symmetric solution enforces the integrality gap be at least $\Omega(\sqrt{n})$ are not longer feasible as the new linear relaxation values are different from the optimum by at most a polylogarithmic factor.

6.2 Tree drawing formulation

In this section we introduce a brand-new ILP formulation for the treewidth problem purely based on geometry of tree-decomposition. The problem is formulated as a network design problem on a grid and the underlying integer linear programming formulation describes the geometric properties of the optimal tree decomposition.

It is known that for any graph $G = (V, E)$ on n vertices there is a binary tree on $\mathrm{O}(n)$ vertices representing the optimal tree decomposition of G. Without loss of generality, assume that such a binary optimal tree decomposition of G on n vertices is given. By Crescenzi et al. [34] any binary tree on n vertices can be properly embedded in the $n \times (\log n + 1)$ grid in the right-down fashion, thus the optimal binary tree decomposition also has such drawing. Hence, we make all further drawings on the $n \times (\log n + 1)$ grid (N, A) with the node set N and the edge set A. We assume that the horizontal grid layers are numbered $\{0, \dots, n-1\}$ in the top-to-bottom manner and the vertical layers are numbered $\{0, \dots, \log n\}$ in the left-to-right manner. Let $y = (i, j)$ be the grid node lying on the horizontal layer i and vertical layer j. We denote y_r, y_u, y_ℓ and y_b the right, upper, left and bottom neighbors of y respectively.

Definition 6.2.1. *For a pair of grid nodes $y = (i, j)$ and $y' = (i', j')$, we write $y \preceq y'$ if $i \leq i'$ and $j \leq j'$. Consistently, we write $y \prec y'$ if $y \preceq y'$ and $y \neq y'$.*

For every vertex $v \in V$ we construct a connected tree T_v which is a subtree of the optimal tree decomposition where every node contains vertex v. T_v is embedded on the edges of the grid such that there is no node in the tree which is adjacent simultaneously to its upper and left neighbors. We call such a tree drawing the *proper right-down drawing*. Notice that in any tree drawn in proper right-down way, there is always a unique left upper node of the tree.

By constraints of the integer linear program below we guarantee that the combined drawing of all vertex subtrees does not contain any cycles, i.e. it is a tree, and for any two adjacent vertices in G the corresponding vertex subtrees are overlapping, i.e. there is a bag of the optimal tree decomposition which contains both vertices.

To model the proper right-down drawing of the vertex subtrees, for every $v \in V$, $e \in A$ and $y \in N$ we introduce the following binary variables:

$$s_{vy} = \begin{cases} 1, & \text{if } y \text{ is the unique left upper corner of } T_v, \\ 0, & \text{otherwise.} \end{cases}$$

$$x_{ve} = \begin{cases} 1, & \text{if } e \in E(T_v), \\ 0, & \text{otherwise.} \end{cases}$$

$$u_{vy} = \begin{cases} 1, & \text{if } y \in V(T_v), \\ 0, & \text{otherwise.} \end{cases}$$

Constraints which model the problem are the following. Existence of a unique left upper corner of the subtree can be straightforwardly described by the equation

$$\sum_{y \in N} s_{vy} = 1, \qquad \forall v \in V. \tag{6.25}$$

Next constraints ensure that in the drawing there is no node y simultaneously adjacent to its left and upper neighbors:

$$x_{v(y_\ell,y)} + x_{u(y,y_u)} \leq 1, \qquad \forall v,u \in V, y \in N. \tag{6.26}$$

Connectivity of the subtrees we guarantee by the constraints

$$u_{vy} = s_{vy} + x_{v(y_\ell,y)} + x_{v(y,y_u)}, \qquad \forall v \in V, y \in N, \tag{6.27}$$

$$x_{v(y,y_r)} \leq u_{vy}, \qquad \forall v \in V, y \in N, \tag{6.28}$$

$$x_{v(y_b,y)} \leq u_{vy}, \qquad \forall v \in V, y \in N. \tag{6.29}$$

Now, when the subtrees are modeled, we set up the inequalities which ensure that for any two adjacent vertices in G the corresponding two subtrees are overlapping. To do this, we notice that in any proper right-down drawing two subtrees are overlapping if and only if the upper-left corner of one of the subtrees belongs to another subtree.

Thus, we complete the construction of the treewidth polytope by setting the following inequalities:

$$s_{uy} + \sum_{y' \preceq y} s_{vy'} \leq 1 + u_{vy}, \qquad \forall (v,u) \in E, y \in N, \tag{6.30}$$

$$s_{uy} + \sum_{y' \in N : y' \not\preceq y, y \not\preceq y'} s_{vy'} \leq 1, \qquad \forall (v,u) \in E, y \in N. \tag{6.31}$$

The integrality constraints are:

$$s_{vy},\ x_{ve},\ u_{vy}\ \in\ \{0,1\}. \tag{6.32}$$

To put the problem in an optimization framework, we add one more variable w for the treewidth of G and require

$$w + 1 \geq \sum_{v \in V} u_{vy}, \qquad \forall y \in N. \tag{6.33}$$

The objective function is given straightforwardly:

$$\min w. \tag{6.34}$$

We refer to the tree drawing ILP formulation as TDF. There is the following relation between the solution of TDF and the treewidth defining vertex elimination order. We find the partial order of vertices using the following proposition and linearize the partial order to obtain an elimination scheme:

Proposition 6.2.2. *Let π be the partial order of vertices in G such that $\pi(v) \leq \pi(u)$ if $y \prec y'$ for $s_{vy} = 1$ and $s_{uy'} = 1$ in the optimal solution to (6.25)–(6.34). Then π is a treewidth defining vertex elimination order for graph G.*

6.2.1 Introduction to local branching

In this subsection we give an overview of the local branching technique introduced by Fischetti and Lodi [46]. The procedure is in the spirit of well-known local search metaheuristics, but the neighborhoods are obtained through the introduction into the Mixed Integer Programming (MIP) model the set of completely general linear inequalities, called *local branching cuts*. This allows the use of a general-purpose MIP solver as a black-box "tactical" tool to explore effectively suitable solution subspaces defined and controlled at a "strategic" level by a simple external branching framework.

We consider a generic MIP of the form:

$$\min c^T x \tag{6.35}$$

subject to

$$Ax \geq b, \tag{6.36}$$

$$x_j \in \{0,1\}, \quad \forall j \in \mathcal{B} \neq \emptyset, \tag{6.37}$$

$$x_j \geq 0, x_j \in \mathbf{Z}, \quad \forall j \in \mathcal{G}, \tag{6.38}$$

$$x_j \geq 0, \quad \forall j \in \mathcal{C}. \tag{6.39}$$

Here, the variable index set $\mathcal{N}$ is partitioned into $(\mathcal{B}, \mathcal{G}, \mathcal{C})$, where $\mathcal{B} \neq \emptyset$ is the index set of the binary variables, while the possibly empty sets $\mathcal{G}$ and $\mathcal{C}$ index the general integer and the continuous variables, respectively. Given a feasible reference solution $\bar{x}$ of the problem, let $\bar{S} = \{j \in \mathcal{B} \mid \bar{x}_j = 1\}$ denote the binary support of $\bar{x}$. For a given positive integer parameter k, we define the k-OPT neighborhood $N(\bar{x}, k)$ of $\bar{x}$ as the set of the feasible solutions satisfying the additional local branching constraint, see [46]:

$$\Delta(x, \bar{x}) = \sum_{j \in \bar{S}} (1 - x_j) + \sum_{j \in \mathcal{B} \setminus \bar{S}} x_j \leq k, \tag{6.40}$$

where the two terms in left-hand side count the number of binary variables flipping their value (with respect to $\bar{x}$) either from 1 to 0 or from 0 to 1, respectively.

The local branching constraint can be used as a branching criterion within an enumerative scheme for MIP: given a solution $\bar{x}$, the solution space associated with the current branching node can be partitioned by means of the disjunction, see [46]:

$$\Delta(x, \bar{x}) \leq k \text{ left branch} \quad \textbf{or} \quad \Delta(x, \bar{x}) \geq k + 1 \text{ right branch}. \tag{6.41}$$

The idea is that the neighborhood $N(\bar{x}, k)$ corresponding to the left branch must be "sufficiently small" to be optimized within short computing time, but still "large enough" to likely contain better solutions than $\bar{x}$. According to the computational results [46], the choice of k in range [10, 20] was effective in most cases.

6.2.2 Local branching technique for TDF

In this subsection we apply the local branching technique [46] to TDF for the treewidth.

The intuition for the local branching method applied to the treewidth is as follows. We start with some (not optimal) tree decomposition T_0 which is a combination of the vertex subtrees T_v in the grid. We define the neighborhood of T_0 as the set of tree decompositions where at most k vertex subtrees T_v might have the left-upper corners different from their locations in T_0. By a MIP solver or by any other suitable method, we find in the constructed neighborhood a tree decomposition T_1 having the minimal treewidth. Making clear the connection to the vertex elimination orders announced in Proposition 6.2.2, this step of the local branching corresponds to finding the (locally) optimal tree decomposition obtained by shifting at most k vertices in a given vertex elimination order. Now, given T_1, we define its neighborhood by the set of tree decompositions where at least $k+1$ subtrees are different from the subtrees in T_0 and at most k subtrees are different from the subtrees in T_1. This can be done by adding two inequalities to the integer linear program. To the extended ILP, we again find a tree decomposition T_2 having the minimal treewidth. We recurse on the integer linear programs, i.e. at step i we obtain a local optimum T_i adding to the current ILP:

- inequalities determining a small neighborhood of T_{i-1},
- inequalities cutting off the neighborhood of T_{i-2}.

We stop the process when the extended ILP does not have any solution. Since the algorithm starts with the whole set of feasible solutions and proceeds till the empty set, the best found solution is a provable global optimum. Below we give a detailed description of the algorithm.

Algorithm 6.1: LocalBranching

```
Generate any feasible $(s_0^*, x_0^*, u_0^*, w_0^*)$.

Generate $(s_1^*, x_1^*, u_1^*, w_1^*)$ by adding the constraint $\Delta(s, s_0^*) \leq k$ to (6.14)–(6.24) and
solving the resulting ILP by a MIP-solver.

Let $i := 2$.

**While** The set of feasible solutions of the current ILP is not empty **Do**
    In the current ILP, replace $\Delta(s, s_{i-2}^*) \leq k$ by $\Delta(s, s_{i-2}^*) \geq k+1$
        and add $\Delta(s, s_{i-1}^*) \leq k$.
    Solve the ILP by MIP to generate $(s_i^*, x_i^*, u_i^*, w_i^*)$.
    Set $i := i + 1$.
**End While**

Output the solution $(s^*, x^*, u^*, w^*)$ found during the previous steps with the
minimum value $w^*$.
```

Given a feasible integer *reference solution*(s^*, x^*, u^*, w^*) to (6.25)–(6.34), let $S^* := \{(v, y) : s_{vy}^* = 1\}$ denote the binary support of the solution. For a given positive integer parameter k, we define the k-optimal neighborhood $\mathcal{N}(s^*, x^*, u^*, w^*, k)$ of (s^*, x^*, u^*, w^*) as the set of feasible solutions to the integer program (6.25)–(6.34) satisfying the additional *local branching constraint*

$$\Delta(s, s^*) := \sum_{(v,y)\in S^*} (1 - s_{vy}) + \sum_{(v,y)\notin S^*} s_{vy} \leq k, \tag{6.42}$$

where the two terms in the left-hand side count the number of binary variables s flipping their value with respect to s^*.

To prune the algorithm, preprocessing rules reducing the size of the graph and constructing safe separators must be applied, see [19, 20]. Clearly, we can easily improve the stop-criteria of the algorithm: if at some step of the algorithm the optimal solution of the linear relaxation to the current ILP is greater than the width of the best found tree decomposition then we can stop since further adding inequalities will only increase the width of the decompositions.

6.3 Comparison of formulations and methods

We start the comparison of the ILP formulations EOF and TDF with the program sizes. Given graph G on n vertices TDF has $O(n^2 \log n)$ variables, while EOF needs $O(n^4)$ variables. The situation is opposite with respect to the number of constraints: TDF contains $O(n^3 \log n)$ constraints, EOF has $O(n^3)$ constraints. Hence, size-wise TDF is slightly more compact than EOF.

It is noticeable that despite its generality, the local branching technique of Fischetti and Lodi [46] is hardly applicable to linear ordering polytopes in general and to EOF in particular. This is due to the fact that all variables in the linear ordering polytopes are related to each other, e.g. shift of one element in the linear order implies changes in $n/2$ variables on average. Moreover, the choice on which variable to branch is also very difficult. This was one of the basic reasons to introduce TDF with variables allowing more freedom for local changes and which are more suitable for the local branching procedure. In the local branching, if k is not very large and only deviations of k elements in the current integer solution are allowed, the spread of the fractional values is not very high and the local optimum can be found quickly.

On the other hand, EOF can be nicely placed within the flow metric framework as it already assumes some "path-like" structure which is the basis of the flow approach. The "flow metric" constraints polish the linear ordering polytope in the neighborhood of the optimal solution which provides better approximation guarantees and speeds up the search. At the same time it is difficult to find a straightforward way to integrate the flow metric approach into TDF as it is based on geometrical and structural properties of the tree-decomposition more than on ordering properties.

It is known that linear ordering formulations, like EOF, work quite well on dense graphs as finding a clique leads to a good progress of the objective function bounds;

but are typically bad on sparse graphs. The situation is somewhat opposite for TDF. It can get stuck in some node of the branching tree if input graph is a clique. Furthermore, the arbitrary spread of k trees T_v which are allowed to be changed is very bad for the input clique graph as the optimal tree-decomposition is just one bag. But this is also the reason for the more effective search on the sparse graphs as we allow the big “jumps” and changes in one step of the algorithm.

6.4 Conclusions and open questions

We considered two Integer Linear Programming formulations for treewidth. The first one is based on the vertex elimination order. It is merged with the flow metric approach by Bornstein and Vempala [24] to obtain better approximation bounds. We show that using this formulation one can obtain straightforwardly a polylogarithmic approximation for the treewidth. The second ILP is structural and it is based on the drawing of the optimal tree decomposition on the grid. This representation leads to some nice geometric properties which we exploit in a local branching procedure proposed by Fischetti and Lodi [46] to obtain an exact algorithm for the Minimum Treewidth problem. The computational experiments on different graph classes can bring more insights about the weaknesses and strengths of the proposed formulations. This is left for further research.

Part III

Treewidth and Applications

Chapter 7

H-Subgraph Edge Deletion

In this chapter of the thesis, we shift our focus to an application of treewidth and tree decompositions. We demonstrate how an otherwise intractable graph theoretical problem can be solved in linear time on graphs that have bounded treewidth, if a tree decomposition of the graph is available.

In the field of combinatorial graph theory, lately there has been an increased interest in algorithms for graphs not containing some specified subgraph or induced subgraph. An apparent reason for this interest is that many combinatorial problems on graphs have useful structural and/or algorithmic properties when restricted to graphs that exclude certain subgraphs. Just to give a few insightful examples: a famous result by Erdös, Kleitman and Rothschild [42] says that almost every triangle-free graph has chromatic number 2; Minty [69] and Sbihi [85] show that the maximum independent set problem is polynomially solvable on claw-free graphs; Dunbar and Frick [41] show that the path-partition conjecture is true for claw-free graphs; Borodin et al. [25] prove that planar graphs with no cycles of length from 4 to 7 are 3-colorable. There are, of course, many other structural and algorithmic results related to graphs without (a) specific subgraph(s). Motivated by this increased attention in the literature, we present algorithms in this chapter to turn a graph into a graph that excludes certain subgraphs by deleting a minimum number of edges from it.

The chapter is organized as follows. In Section 7.1 we introduce the problem that will be the subject of study in this chapter and we give some references to related studies. We continue by showing in Section 7.2 that by a general result of Courcelle [31, 32], the decision version of the considered problem is theoretically solvable in linear time on graphs of bounded treewidth. We then introduce a dynamic program in Section 7.4 for the case where the input graph has bounded maximum degree that solves the problem in linear time on graphs of bounded treewidth. In Section 7.4 the subgraph that is excluded from the graph can be any fixed, connected graph. Subsequently in Section 7.5, we consider the case where the fixed subgraph forms a clique and we present a dynamic program that solves the problem in linear time on graphs of bounded treewidth. In Section 7.6, we present Baker's style approximation schemes for the problems from Sections 7.4 and 7.5 on planar graphs. Finally, in Section 7.7, we show how our algorithms can be adopted to deal with the situation where more than one type of subgraph needs to be excluded from the input graph.

7.1 Problem definition

We consider the graph theoretical problem of turning a graph into a graph that excludes certain subgraphs. The transformation is executed by removing edges from the input graph. Given an input graph G and a finite set of graphs $\mathcal{H} = \{H_1, \ldots, H_t\}$, we call a subgraph of G an H-subgraph of G if it is isomorphic to one of the graphs in the set $\mathcal{H}$. In this context, a graph G is usually referred to as the *text* and $\mathcal{H}$ as the set of *patterns*. A graph G is called *H-free* if there are no H-subgraphs in G. For ease of notation, we use notions of H-subgraphs and H-free graphs instead of $\mathcal{H}$-subgraphs and $\mathcal{H}$-free graphs.

We restrict ourselves to the case in which the subgraphs that have to be excluded are fixed graphs that are connected. From here on, we will therefore assume that $\mathcal{H}$ forms a finite set of fixed, connected patterns H_i. The optimization problem that we consider in this chapter is the following.

PROBLEM:	MINIMUM H-SUBGRAPH EDGE DELETION
Input:	Text graph G and finite set of patterns $\mathcal{H} = \{H_1, \ldots, H_t\}$
Question:	What is the minimum number of edges that must be deleted from G to make it H-free?

We refer to the decision version of this problem as H-SUBGRAPH EDGE DELETION; i.e., given integer k, is it possible to make the input graph H-free by deleting at most k edges?

Related work. It is well-known that the TRIANGLE EDGE DELETION problem, a special case of H-SUBGRAPH EDGE DELETION with $\mathcal{H} = \{K_3\}$, is *NP*-complete, see Yannakakis [94]. Therefore the problem H-SUBGRAPH EDGE DELETION is *NP*-complete as well. Recently, Brügmann et al. [26] proved that the problem TRIANGLE EDGE DELETION remains *NP*-complete even if the graph is planar and has maximum degree of 7. On the positive side, they construct polynomial time reduction rules to obtain linear problem kernels.

Closely related problems are considered extensively in the extremal graph theory; see, e.g., Bollobás [22] and Bohman [21]. There, the general question is: Given a graph H and a number n, what is the maximum number of edges in a graph G on n vertices that does not contain a subgraph isomorphic to H? In the context of this paper, the bounds obtained in the extremal graph theory can be seen as the source of generic lower bounds on the number of edges to be removed to make a given graph H-free.

To construct PTASs for MINIMUM H-SUBGRAPH EDGE DELETION on planar graphs, we employ the layerwise decomposition approach introduced by Baker [6]. A generalization of this approach that can be applied also to problems with a non-local structure has recently been developed by Demaine and Hajiaghayi and is an extension of the bidimensionality theory. For a literature overview on bidimensionality and its connections to approximation schemes for planar graph problems, we refer to [37] and [38].

Our results. In this chapter, we first implicitly present a linear time algorithm for H-SUBGRAPH EDGE DELETION on graphs of bounded treewidth, based on a framework by Courcelle [31, 32]. We then introduce constructive linear time algorithms for MINIMUM H-SUBGRAPH EDGE DELETION for the two cases for which respectively G has bounded maximum vertex degree and H is a clique. Both these algorithms are dynamic programs that assume as input a tree decomposition of the input graph. Finally, for the same cases we design PTASs for MINIMUM H-SUBGRAPH EDGE DELETION on planar graphs.

7.2 MSOL Formulations

Bounded treewidth and Monadic Second Order Logic (MSOL) have proven to be key concepts in establishing fixed-parameter tractability results. The general results of Courcelle [31, 32] and Arnborg et al [4] provide a host of powerful algorithmic tools for many combinatorial problems on graphs of bounded treewidth. In particular, they show that any property on graph G that can be expressed in MSOL, can be decided in linear time if G has bounded treewidth.

7.2.1 Formulation for single pattern

Consider the H-SUBGRAPH EDGE DELETION problem where $\mathcal{H}$ consists of a single pattern H. By encoding this problem in MSOL, we implicitly construct a linear time algorithm solving H-SUBGRAPH EDGE DELETION on graphs of bounded treewidth.

We end the section by generalizing the results to allow for a finite set $\mathcal{H}$ of patterns instead of just a single pattern.

Consider the following relational structure:

$$\mathbf{G} = \{V(G), E(G), V(H), E(H), R_G^2, R_H^2, Eq^2, f^1\},$$

where

- $R_G(x,e)$ and $R_H(x,e)$ are the vertex-edge incidence relation in G and H respectively;
- $Eq(x,y)$ is the equality predicate;
- $f : S \to V(H)$ is a function with domain $S \subseteq V(G)$ and image $V(H)$.

We now describe a formula $\Phi(k,G,H)$ such that $\mathbf{G} \models \Phi(k,G,H)$ if and only if there exists a set of edges $F \subseteq E(G)$ of cardinality at most k covering all H-subgraphs of the graph G.

Let G' be a subgraph of G. First, we define the formula $\Psi(f,G')$ to express that f is an isomorphism between G' and H. The following properties should be satisfied:

- Function f is injective:

$$Inj(f,G') \;=\; \forall\, u,v \in V(G'),\; Eq(f(u),f(v)) \Rightarrow Eq(u,v);$$

- Function f is surjective:

$$Surj(f,G') \;=\; \forall\, v \in V(H)\; \exists\, u \in V(G'),\; Eq(f(u),v);$$

- Function f preserves the edge relations from G' in H:

$$\begin{aligned}\overrightarrow{Edge}(f,G') \;=\;& \forall\, u,v \in V(G'), e \in E(G'),\; R_G(u,e) \,\&\, R_G(v,e) \Rightarrow \\ & \exists\, e' \in E(H),\; R_H(f(u),e') \,\&\, R_H(f(v),e');\end{aligned}$$

- Function f preserves the edge relation from H in G':

$$\begin{aligned}\overleftarrow{Edge}(f,G') \;=\;& \forall\, u,v \in V(H), e \in E(H),\; R_H(u,e) \,\&\, R_H(v,e) \Rightarrow \\ & \exists\, u',v' \in V(G'),\; e' \in E(G'), \\ & R_G(u',e') \,\&\, R_G(v',e') \,\&\, Eq(f(u'),u) \,\&\, Eq(f(v'),v).\end{aligned}$$

Let

$$\Psi(f, G') = Inj(f, G') \,\&\, Surj(f, G') \,\&\, \overrightarrow{Edge}(f, G') \,\&\, \overleftarrow{Edge}(f, G'),$$

and consequently

$$\begin{aligned}\Phi(k, G, H) \;=\; & \exists F \subseteq E(G), |F| \leq k \,\& \\ & (\forall V' \subseteq V(G), E' \subseteq E(G),\ \Psi(f, (V', E')) \Rightarrow |F \cap E'| \geq 1).\end{aligned}$$

Hence, we expressed the problem H-SUBGRAPH EDGE DELETION in MSOL, which in combination with Courcelle's Theorem proves the following theorem.

Theorem 7.2.1. *Problem H-*SUBGRAPH EDGE DELETION *with $\mathcal{H}$ consisting of a single fixed, connected pattern H can be solved in $O(n \cdot g(w, h))$ time for some function g, where w is the treewidth of G and h is the size (number of vertices) of H.*

Using the result of Theorem 7.2.1 in combination with binary search, the corresponding optimization version of the problem can be solved in $\mathrm{O}(n \log(m) \cdot g(w, h))$ time, where m denotes the number of edges in G.

7.2.2 Formulation for set of patterns

To generalize Theorem 7.2.1 to a finite set of patterns $\mathcal{H} = \{H_1, \ldots, H_t\}$, we simply need a wider relational structure **G**:

$$\mathbf{G} = \{V(G), E(G), R_G, Eq\} \cup_{i=1}^{t} \{V(H_i), E(H_i), R_i, f_i\},$$

where in addition to the definitions above we have to specify

- $R_i(x, e), i = 1 \ldots t$, are the vertex-edge incidence relations in H_i;
- $f_i : S \subseteq V(G) \to V(H_i), i = 1 \ldots t$, are functions with domain $S \subseteq V(G)$ and images $V(H_i)$.

Then, we can write down the general formula $\Phi(k, G, \mathcal{H})$ in the following way:

$$\begin{aligned}\Phi(k, G, \mathcal{H}) \;=\; & \exists F \subseteq E(G), |F| \leq k \,\&(\forall V' \subseteq V(G), E' \subseteq E(G), \\ & \exists i \in \{1, \ldots t\},\ \Psi(f_i, (V', E')) \Rightarrow |F \cap E'| \geq 1).\end{aligned}$$

This MSOL formulation, in combination with the Courcelle's Theorem, proves the following theorem.

Theorem 7.2.2. *Problem H-*SUBGRAPH EDGE DELETION *with a finite set of fixed, connected patterns $\mathcal{H} = \{H_1, \ldots, H_t\}$ can be solved in $O(n \cdot g(w, h))$ time for some function g, where w is the treewidth of G and h is the size of the largest pattern in $\mathcal{H}$.*

To solve the corresponding optimization version of the problem, one can combine Theorem 7.2.2 with binary search, which adds a factor $\log(m)$ to the complexity.

7.3 Nice tree decompositions

A tree decomposition (T, X) is called a *nice tree decomposition* if the following conditions are satisfied: Every node of the tree T has at most two children; if a node i has two children j and k, then $X_i = X_j = X_k$; and if a node i has one child j, then either $|X_i| = |X_j| + 1$ and $X_j \subset X_i$ or $|X_i| = |X_j| - 1$ and $X_i \subset X_j$. The following result is from [64]:

Lemma 7.3.1. *A tree decomposition (T, X) of a graph G can be transformed without increasing its width into a nice tree decomposition of G in time polynomial in $|X|$ and the size of G. The size of the resulting nice tree decomposition is $O(w|X|)$, where w is the width of the tree decomposition.*

We point out that any tree decomposition of G can be transformed into a tree decomposition of the same width having at most n bags by simply removing all bags that are subset of another bag. Hence using Fact 7.3.1 we assume from here on that our algorithms run on nice tree decompositions with $O(wn)$ bags, rooted in some arbitrary bag X_r.

7.4 DP for text graphs with bounded degree

In this section, we consider problem MINIMUM H-SUBGRAPH EDGE DELETION in the setting where text graph G has bounded degree and bounded treewidth and $\mathcal{H}$ consists of a single fixed, connected pattern H. We construct a dynamic programming algorithm for this setting that solves MINIMUM H-SUBGRAPH EDGE DELETION in time that is exponential only in the maximum degree of G, in the width of a tree decomposition of G and in some fixed parameters of H. A generalization of this result to the setting where $\mathcal{H}$ is a finite set of fixed, connected patterns will be introduced in Section 7.7.

7.4.1 Constant parameters and notation

By w, we denote the *width* of (T, X). The *maximum degree* in G will be denoted by $\Delta(G)$. The *diameter* of a graph $G = (V, E)$ is the maximum over all vertex pairs $\{u, v\} \subseteq V$ of the length of a shortest path between u and v in G. By h and d we denote respectively the *size* $|H|$ and the diameter of pattern H. Since G has bounded degree and bounded treewidth and H is a fixed connected pattern, the values $\Delta(G)$, w, h and d are all constants in the dynamic program.

Given a bag X in a nice tree decomposition (T, X), we let T_X be a subtree of T that is rooted in X and we let $G[T_X]$ be the subgraph of G that is induced by all vertices from the bags in T_X. By $\mathcal{E}_X$ we denote the set of all edges of G for which both end points are present in bag X and by E_X, we denote a subset of $\mathcal{E}_X$, i.e., $E_X \in 2^{\mathcal{E}_X}$. For a bag X, by V_X we denote the set of vertices of G that are in bag X. For a subtree T_X, by V_{T_X} we denote the set of vertices of G that are present in some bag of T_X. Finally for a vertex set S, by $\mathcal{H}_S$ we denote the set of different H-subgraphs in G that are incident to some vertex in S and by H_S, we denote a subset of $\mathcal{H}_S$.

Lemma 7.4.1. *Consider a graph G and a tree decomposition (T, X) of G of width w. Let $Q = h!\binom{\Delta(G)^{d+1}}{h}$, then*

- *for vertex v, $|\mathcal{H}_{\{v\}}|$ is bounded from above by Q, and*
- *for bag X in (T, X), the value $|\mathcal{H}_{V_X}|$ is bounded from above by $(w+1)Q$.*

Proof. Given that v corresponds to some vertex of an H-subgraph in G it is clear that all h vertices in this H-subgraph have distance at most d to v in this H-subgraph and thus also in G. Since the maximum degree in G is $\Delta(G) > 1$, there are at most $\sum_{i=0}^{d} \Delta(G)^i = \frac{\Delta(G)^{d+1}-1}{\Delta(G)-1} \leq \Delta(G)^{d+1}$ vertices in G that have distance at most d to v. Since $\binom{\Delta(G)^{d+1}}{h}$ different sets of h vertices can be chosen among $\Delta(G)^{d+1}$ vertices and each such set can contain at most $h!$ different H-subgraphs, the result follows. The second statement is a straightforward corollary of the first one, since $|X| \leq w+1$. □

7.4.2 Dynamic program and results

First we note that by deleting a set of edges from G that covers all H-subgraphs of G, G becomes H-free. Given a subtree T_X of T, we define:

- $F(E_X, T_X, H_{V_X})$ is the minimum cardinality of a set S of edges from $G[T_X]$ covering all H-subgraphs from $(\mathcal{H}_{V_{T_X}} \setminus \mathcal{H}_{V_X}) \cup H_{V_X}$ in G, given that $S \cap \mathcal{E}_X = E_X$
- $F(E_X, T_X, H_{V_X}) = \infty$ if H-subgraphs from $(\mathcal{H}_{V_{T_X}} \setminus \mathcal{H}_{V_X}) \cup H_{V_X}$ can *not* be covered by E_X plus some set of edges from $E(G[T_X]) \setminus \mathcal{E}_X$.

For each bag X in (T, X), we compute a table of such $F-$values for subtrees rooted in X, one value for each combination of a subset from $\mathcal{E}_X$ and a subset from $\mathcal{H}_{V_X}$. One such table thus contains $2^{|\mathcal{E}_X|+|\mathcal{H}_{V_X}|}$ values. To be more specific, we compute tables for the following three types of subtrees:

1. For each bag X in T, except for root bag X_r, we compute the table for the subtree consisting of the parent X^+ of X, X and all descendants of X in T. The root bag of this subtree is X^+. Such subtree will be denoted by T_{X^+}.

2. For each bag X with two children we compute a table for the subtree T_X, consisting of bag X and all descendants of X.

3. For each leaf bag X we compute a table for the subtree T_X.

Note that for a leaf bag X, subtree T_X equals bag X. Therefore the tables for the subtrees of type 3 are easy to compute.

Lemma 7.4.2. *A value in the table of the subtree T_X rooted in a leaf bag can be computed in constant time.*

Proof. To compute one value $F(E_X, T_X, H_{V_X})$ for the table of the subtree T_X that is rooted in leaf bag X, one just needs to check whether edge set E_X covers all H-subgraphs of H_{V_X}. If this is the case, then $F(E_X, T_X, H_{V_X}) = |E_X|$, else $F(E_X, T_X, H_{V_X}) = \infty$. Using the observation that $|E_X| \leq w^2$, Lemma 7.4.1 and the fact that both w and Q are constants, the result follows. $\square$

The following lemmas give recursive formulas that show how to compute the tables for subtrees that are rooted in a non-leaf bag of tree decomposition (T, X). First we show how to update the table when we combine two subtrees that are rooted in the same bag and have only this bag in common.

Lemma 7.4.3. *Let T'_X and T''_X be two subtrees rooted in X that only share bag X. Let T_X be the subtree that is obtained by combining T'_X and T''_X and let $H_{V_X}, I_{V_X}, J_{V_X} \in 2^{\mathcal{H}_{V_X}}$. Then*

$$F(E_X, T_X, H_{V_X}) = \min_{\substack{I_{V_X}, J_{V_X}: \\ H_{V_X} \subseteq I_{V_X} \cup J_{V_X}}} F(E_X, T'_X, I_{V_X}) + F(E_X, T''_X, J_{V_X}) - |E_X|.$$

The next two lemmas show how to update the values for a subtree when we extend it by the parent bag of its root.

Lemma 7.4.4. *Let T_Y be a subtree rooted in Y and let $X = Y \cup \{v\}$ be the parent bag of Y. Furthermore let $T_X = T_{Y^+}$ and let $E_Y = E_X \cap \mathcal{E}_Y$. Then:*

$$F(E_X, T_X, H_{V_X}) = \min_{\substack{H_{V_Y}:\ E_X \setminus E_Y \text{ covers} \\ H_{V_X} \setminus H_{V_Y} \text{ in } G}} F(E_Y, T_Y, H_{V_Y}) + |E_X \setminus E_Y|$$

and $F(E_X, T_X, H_{V_X}) = \infty$ if there is no such H_{V_Y}.

Lemma 7.4.5. *Let T_Y be a subtree rooted in Y and let $X = Y \setminus \{v\}$ be the parent bag of Y. Furthermore, let $T_X = T_{Y^+}$. Then:*

$$F(E_X, T_X, H_{V_X}) = \min_{\substack{E_Y, H_{V_Y}:\ H_{V_X} \cup (\mathcal{H}_{\{v\}} \setminus \mathcal{H}_{V_X}) \subseteq H_{V_Y}, \\ E_Y \cap \mathcal{E}_X = E_X}} F(E_Y, T_Y, H_{V_Y}).$$

Obviously, when bag Y and parent bag X of Y have the same vertex set, then the update of the table after an extension by the parent bag can be accomplished by simply copying the values. The following lemma gives an upper bound on the time to compute one F-value.

Lemma 7.4.6. *The time needed to determine an F-value by one of the Lemmas 7.4.3, 7.4.4 or 7.4.5 is bounded from above by* $O(2^{w^2+2(w+1)Q}w^4Q^2)$.

Proof. To determine a value using Lemma 7.4.3 takes $O(4^{|\mathcal{H}_{V_X}|}|\mathcal{H}_{V_X}|^2)$ time. Using Lemma 7.4.1, this time is bounded from above by the value $O(4^{(w+1)Q}w^2Q^2)$. To determine a value using Lemma 7.4.4, for all subsets H_{V_Y} we have to check whether all elements from $H_{V_X} \setminus H_{V_Y}$ contain an edge from $E_X \setminus E_Y$. This takes less than $2^{|\mathcal{H}_{V_Y}|}|H_{V_X}||E_X|$ time and by using Lemma 7.4.1 and the observation that $|E_X| \leq w^2$, this is bounded from above by $O(2^{(w+1)Q}w^3Q)$. To determine a value using Lemma 7.4.5, for all combinations of E_Y and H_{V_Y} we have to check whether $H_{V_X} \subseteq H_{V_Y}$, whether $\mathcal{H}_{\{v\}} \setminus \mathcal{H}_{V_X} \subseteq H_{V_Y}$ and whether $E_Y \cap \mathcal{E}_X = E_X$. This takes less than $2^{|\mathcal{E}_Y|+|\mathcal{H}_{V_Y}|}(|H_{V_X}| + |\mathcal{H}_{V_X}|^2 + |E_X|^2)$ time and as shown before this can be bounded by $O(2^{w^2+(w+1)Q}w^4Q^2)$. Clearly, all these running times are smaller than $O(2^{w^2+2(w+1)Q}w^4Q^2)$. □

Using Lemmas 7.4.2, 7.4.3, 7.4.4 and 7.4.5, we can compute all necessary tables. For all bags in the tree, in post order, we construct the tables for the subtree(s) rooted in this bag. The proof of Lemma 7.4.2 shows how to construct the table for a subtree rooted in leaf bag X. If X is not a leaf bag, suppose X has 1 child Y, then we may assume that earlier already we computed the table for subtree T_Y. By using Lemma 7.4.4 or 7.4.5 we then can compute the table for subtree $T_X = T_{Y^+}$ from the table of T_Y. If X is not a leaf bag and it has 2 children X and Y, then by definition of a nice tree decomposition $X = Y = Z$ and we may assume that we already computed the tables for subtrees T_Y and T_Z. By simply copying the values we can construct the tables for subtrees T_{Y^+} and T_{Z^+} from the tables of T_Y respectively T_Z. Then we use Lemma 7.4.3 to compute the table for subtree T_X from the tables for subtrees T_{Y^+} and T_{Z^+}. The answer to MINIMUM H-SUBGRAPH EDGE DELETION can be found in the table of T_{X_r}. To be more precise, the solution is

$$\min_{E_{X_r}} F(E_{X_r}, T_{X_r}, \mathcal{H}_{V_{X_r}}).$$

Using the optimal set E_{X_r}, one can perform a backward search in the tree to determine a minimum set of edges that must be deleted from G to make it H-free. To estimate the running time of the dynamic program, in the following lemma we determine an upper bound on the number of individual F-values that must be computed:

Lemma 7.4.7. *In the dynamic program, at most $O(n2^{w^2+(w+1)Q}w)$ F-values need to be determined.*

Proof. A bag in the rooted tree T has at most 2 children. Thus for any bag X, we compute a table for at most 3 subtrees rooted in bag X, namely T_X and T_{Y^+}, T_{Z^+} for possible children Y, Z of X. We recall that the number of bags in T is bounded by $\mathrm{O}(wn)$. Therefore the total number of tables to construct is bounded by $\mathrm{O}(wn)$. Since the width of (T, X) is w, there are at most $w+1$ vertices in bag X and therefore $|\mathcal{E}_X|$ is bounded from above by $\frac{w^2+w}{2} \leq w^2$ for $w \geq 1$. Furthermore, by Lemma 7.4.1, $|\mathcal{H}_{V_X}|$ is bounded from above by $(w+1)Q$. The latter two observations imply that the number of values in one table is bounded from above by $2^{w^2+(w+1)Q}$, from which the result follows. □

The main result of this section is described in the following theorem:

Theorem 7.4.8. *Given a graph G of bounded maximum degree $\Delta(G)$, a fixed connected pattern H and a nice tree decomposition (T, X) of G of bounded width w. The problem* MINIMUM H-SUBGRAPH EDGE DELETION *on G with $\mathcal{H} = \{H\}$ can be solved in time $O(n2^{2w^2+3(w+1)Q}w^5Q^2)$, where $Q = h!\binom{\Delta(G)^{d+1}}{h}$.*

Proof. By Lemma 7.4.7, we need to compute at most $\mathrm{O}(n2^{w^2+(w+1)Q}w)$ F-values in the dynamic program that solves the problem and by Lemma 7.4.6, it takes at most $\mathrm{O}(2^{w^2+2(w+1)Q}w^4Q^2)$ time to compute one such value. Combining these results, we conclude that the dynamic program runs in $\mathrm{O}(n2^{2w^2}w^5\binom{w}{h-1}h^2)$ time. □

7.5 Dynamic program for clique patterns

In this section, we consider MINIMUM H-SUBGRAPH EDGE DELETION for the special case where H is a clique and G has bounded treewidth. Compared to Section 7.5 we thus drop the constraint that G should have bounded vertex degree. We exploit the fact that every tree decomposition contains a bag with all vertices from the clique (see Lemma 2.6.3) and we find a polynomial time algorithm that again acts on a nice tree decomposition of the text graph G. It is important to state here that in this section we consider H-subgraphs of G that are induced by the vertex set of $G[T_X]$, not by the vertex set of G.

7.5.1 Dynamic program and results

For subtree T_X of T rooted in bag X, we define:

- $F(E_X, T_X)$ is the minimum cardinality of a set S of edges from $G[T_X]$ that covers all H-subgraphs of $G[T_X]$, given that $S \cap \mathcal{E}_X = E_X$.
- $F(E_X, T_X) = \infty$, if **not** all H-subgraphs of $G[T_X]$ can be covered by E_X plus some set of edges from $E(G[T_X]) \setminus \mathcal{E}_X$.

We compute tables for the same subtrees as in the previous section. Note that one table for a subtree rooted in bag X now consists of $2^{|\mathcal{E}_X|}$ F-values. Using similar arguments as those used in the previous section, it is easy to prove the following:

Lemma 7.5.1. *During a run of the dynamic program, at most $O(n2^{w^2}w)$ individual F-values have to be determined.*

As in the previous section, a value for the table of a subtree that is rooted in a leaf bag can be computed in constant time. The following lemmas give recursive formulas that show how to compute the tables for subtrees that are rooted in a non-leaf bag of T. The first lemma shows how we can combine subtrees that are rooted in the same bag and have only this bag in common.

Lemma 7.5.2. *Let T_X be obtained by taking the union of subtrees T'_X and T''_X such that the root X of T'_X and T''_X is the only bag that belongs to both subtrees. Then*

$$F(E_X, T_X) \;=\; F(E_X, T'_X) + F(E_X, T''_X) - |E_X|.$$

The next two lemmas show how to update the values for a subtree when we extend it by the parent bag of its root.

Lemma 7.5.3. *Let T_Y be a subtree of T rooted in bag Y, let $X = Y \cup \{v\}$ be the parent bag of Y in T. Furthermore let $T_X = T_{Y^+}$ and let $E_Y = E_X \cap \mathcal{E}_Y$. Then:*

$$F(E_X, T_X) = \begin{cases} F(E_Y, T_Y) + |E_X \setminus E_Y|, & \textit{if } E_X \textit{ covers all H-subgraphs} \\ & \textit{of } G[T_X] \textit{ that are incident to} \\ & \textit{the vertex } v. \\ \infty, & \textit{otherwise.} \end{cases}$$

Indeed, since X is the only bag containing v in T_X, it also contains all neighbors of v in $G[T_X]$. Therefore, all edges of an H-subgraph of $G[T_X]$ that are incident to v are part of $\mathcal{E}_X$ and thus if such an H-subgraph is not covered by E_X then it is not covered at all.

Lemma 7.5.4. *Let T_Y be a subtree of T rooted in bag Y, let $X = Y \setminus \{v\}$ be the parent bag of Y in T and let $T_X = T_{Y^+}$. Then*

$$F(E_X, T_X) = \min_{E_Y \,:\, E_Y \cap \mathcal{E}_X = E_X} F(E_Y, T_Y).$$

Again, when bag Y and parent bag X of Y have the same vertex set, then the update of the table after an extension by the parent bag can be accomplished by simply copying the values. In the previous section, it is explained how Lemmas 7.5.2, 7.5.3 and 7.5.4 can be used to determine the tables for all necessary subtrees. The minimum value in the table of subtree T_{X_r} is the solution to MINIMUM H-SUBGRAPH EDGE DELETION.

Lemma 7.5.5. *The time needed to determine an F-value in the algorithm is bounded from above by $O(2^{w^2} w^4 \binom{w}{h-1} h^2)$.*

Proof. Determining the value for a subtree rooted in a leaf bag or by using Lemma 7.5.2 takes constant time. When using Lemma 7.5.3, we check whether E_X covers all H-subgraphs of $G[T_X]$ that are incident to v. Vertex v has at most w neighbors in $G[T_X]$, so we have to check for each of the at most $\binom{w}{h-1}$ different combinations of $(h-1)$ such neighbors whether they form an h-clique with v in $G[T_X]$ for which all $\frac{h^2-h}{2}$ edges are in $\mathcal{E}_X \setminus E_X$. Since $|\mathcal{E}_X|$ is bounded by $O(w^2)$, this takes at most $O(w^2 \binom{w}{h-1} h^2)$ time. When using Lemma 7.5.4 to determine an F-value, for all E_Y we have to check whether $E_Y \cap \mathcal{E}_X = E_X$, which can be done in time $O(2^{w^2} w^4)$. Clearly, all these algorithmic time complexities are bounded from above by $O(2^{w^2} w^4 \binom{w}{h-1} h^2)$. □

The main result of this section is described in the following theorem:

Theorem 7.5.6. *Given an arbitrary text graph G, clique pattern H of size h and a nice tree decomposition (T, X) of G of bounded width w with N bags. Then* MINIMUM H-SUBGRAPH EDGE DELETION *on graph G with $\mathcal{H} = \{H\}$ can be solved in $O(n2^{2w^2} w^5 \binom{w}{h-1} h^2)$ time.*

Proof. By Lemma 7.5.1, we need to compute at most $O(n2^{w^2} w)$ F-values in the dynamic program that solves the problem and by Lemma 7.5.5, the computation of one such value takes at most $O(2^{w^2} w^4 \binom{w}{h-1} h^2)$ time. Combining these results, we conclude that the dynamic program runs in $O(n2^{2w^2} w^5 \binom{w}{h-1} h^2)$ time. □

7.6 Baker's approximation scheme

In this section, we consider the problem MINIMUM H-SUBGRAPH EDGE DELETION for planar text graphs and patterns. We combine the dynamic programs from Sections 7.4 and 7.4 with a technique invented by Brenda Baker to construct PTASs for the two cases where respectively G has bounded vertex degree and where H is a 3-clique or 4-clique.

7.6.1 Bounded outerplanarity index

The following two lemmas form the analogues to respectively Theorem 7.4.8 and Theorem 7.5.6. The only difference is that they assume planar input graphs with bounded outerplanarity index instead of regular input graphs with bounded treewidth.

Lemma 7.6.1. *Given a planar text graph G of bounded maximum degree $\Delta(G)$ and bounded outerplanarity index l and a fixed connected pattern H, an optimal solution to* MINIMUM H-SUBGRAPH EDGE DELETION *can be obtained in time $O(n2^{18l^2+9lQ}l^5Q^2)$, where $Q = h!\binom{\Delta(G)^{d+1}}{h}$.*

Proof. By Theorem 2.4.2, the outerplanarity index l of G can be determined in $O(n^2)$ time. By Theorem 2.6.6, $tw(G) \leq 3l - 1$ and by Theorem 2.6.5, a tree decomposition of G of width $w \leq 3l - 1$ can be obtained in linear time that can be turned into a nice tree decomposition in linear time. By Theorem 7.4.8 we can use this nice tree decomposition to solve MINIMUM H-SUBGRAPH EDGE DELETION on G in time $O(n2^{2w^2+3(w+1)Q}w^5Q^2)$. Since $w \leq 3l - 1$, the result follows. □

A similar result can be obtained for MINIMUM H-SUBGRAPH EDGE DELETION on planar graphs when H is a 3-clique or 4-clique. Note that to cover all K_2-subgraphs of a graph by edges, one simply has to select as an Edge Cover the complete set of edges. Moreover, note that a planar graph G does not have K_k as a subgraph for $k \geq 5$. This is why we only consider K_3 and K_4 as subgraphs in this section that deals with planar graphs.

Lemma 7.6.2. *Given a planar text graph G of bounded outerplanarity index l and pattern H that is either a K_3 or K_4, an optimal solution to* MINIMUM H-SUBGRAPH EDGE DELETION *on G can be obtained in time $O(n2^{18l^2}l^5\binom{3l-1}{h-1}h^2)$.*

Proof. Similar to the proof of Lemma 7.6.1. By Theorem 7.5.6, MINIMUM H-SUBGRAPH EDGE DELETION on G can be solved in time
$O(n2^{2w^2}w^5\binom{w}{h-1}h^2)$. Since $w \leq 3l - 1$, the result follows. □

The problem MINIMUM H-SUBGRAPH EDGE DELETION on planar graphs of bounded degree with fixed, connected pattern is thus fixed parameter tractable, since it is tractable when parameterized by outerplanarity index l of G. The same holds for MINIMUM H-SUBGRAPH EDGE DELETION on planar graphs with clique pattern. We will use these properties to construct a Baker's approximation scheme in the next section that can be applied to both problems.

7.6.2 Approximation schemes

Given a planar embedding of a planar graph $G = (V, E)$, we say that a vertex v is of *level* 1 if it is on the exterior face of the embedding. Let V_i be the set of all vertices of level i or lower than i, then vertex w is in level $i+1$ if it is on the exterior face of the embedding induced by $G[V \setminus V_i]$. We say that edge $e = (v, w) \in E$ is in level i of the embedding if both vertices v and w are in level i. We assume in this section that a planar embedding is represented by an appropriate data structure such that levels of vertices can be computed in linear time.

In the following two theorems and their proof, we denote by E_{opt} a minimum set of edges from G covering all H-subgraphs of G and by OPT we denote the size of E_{opt}. First we present an approximation scheme for MINIMUM H-SUBGRAPH EDGE DELETION on a planar graph G with bounded degree and arbitrary fixed, connected pattern H. A generalization of the results to finite sets of fixed, connected patterns will be introduced in Section 7.7.

Theorem 7.6.3. *For a planar text graph G of bounded degree, a fixed connected pattern H and any $s > 0$, there is a $O(n^2 2^{18l^2+9lQ} l^5 Q^2)$-time algorithm for* MINIMUM H-SUBGRAPH EDGE DELETION *that finds a solution of size at most $(\frac{s+1}{s})OPT$, where l is a constant depending on s and the outerplanarity index of H.*

Proof. First, we use Theorem 2.4.2 to determine G's outerplanarity index k and a k-outerplanar embedding of G in $O(n^2)$ time. I.e, each vertex of G belongs to one of the k levels. Similarly, we determine H's outerplanarity index k' in $O(h^2)$ time. Since H is a fixed subgraph, k' is a constant. If k was a constant, the problem could be solved in polynomial time by complete enumeration. Therefore it is reasonable to assume that $k > (s+2)k'$. Now for some fixed $2k' \leq l \leq k$ and for each $i \in I = \{mk'+1 \mid 0 \leq m \leq \lfloor \frac{l}{k'} - 2 \rfloor\}$ we construct a set G^i of induced subgraphs of G, consisting of the l-outerplanar subgraphs of G:

- induced by levels 1 to $i + k' - 1$.
- induced by levels $j(l-k') + i$ to $j(l-k') + i + l - 1,\ 0 \leq j \leq \lfloor \frac{k-i-l+1}{l-k'} \rfloor$.
- induced by levels $\lfloor \frac{k-i-l+1}{l-k'} \rfloor (l-k') + i + l - k'$ to k.

See Figure 7.1 for an illustration. Note that $i + k' - 1 \leq \lfloor \frac{l}{k'} - 2 \rfloor k' + 1 + k' - 1 \leq l - k' < l$

and also that $k - (\lfloor \frac{k-i-l+1}{l-k'} \rfloor (l-k') + i + l - k') + 1 \leq k - (k - i - 2l + 1 + k' + i + l - k') + 1 = l$, so both the subgraph induced by the first bullet and the one induced by the third bullet are l-outerplanar. Clearly, also the subgraphs under the second

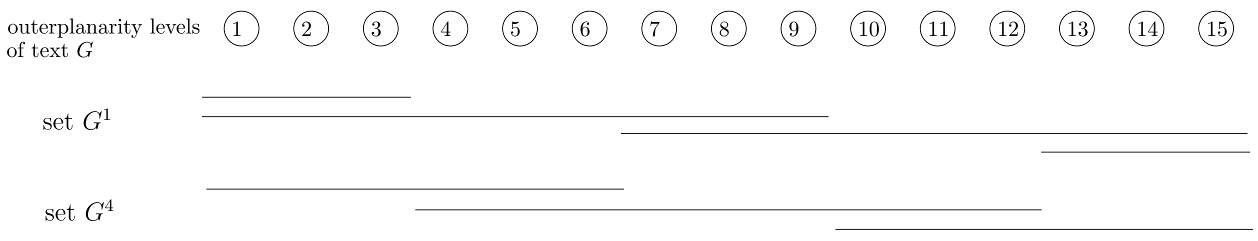


Figure 7.1: Suppose text G has 15 outerplanarity levels and pattern H is 3-outerplanar. For $l = 9$, we construct two sets (G^1 and G^4) of 9-outerplanar subgraphs of G. For example, the first graph in G^1 is induced by all vertices in G's first three levels.

bullet are l-outerplanar. Since $|I| = \lfloor \frac{l}{k'} - 1 \rfloor$ and for each i we construct $\lfloor \frac{k-i-l+1}{l-k'} + 3 \rfloor$ subgraphs, in total we construct less than $(\frac{l-k'}{k'})(\frac{k+2l+1}{l-k'}) = \frac{k+2l+1}{k'} \leq 3k \leq 3n =O(n)$ induced subgraphs of G for fixed value of l.

Observation 7.6.4. *For every $i \in I$, the vertices in the following set of levels are the only vertices that are part of more than one graph from G^i:*

- *levels $j(l-k')+i, \ldots, j(l-k')+i+k'-1$, $0 \leq j \leq \lfloor \frac{k-i-l+1}{l-k'} \rfloor$ and*
- *levels $\lfloor \frac{k-i-l+1}{l-k'} \rfloor (l-k')+i+l-k'$ to $\lfloor \frac{k-i-l+1}{l-k'} \rfloor (l-k')+i+l-1$.*

Observation 7.6.5. *For at least one $i \in I$ the set of levels from Observation 7.6.4 contains at most $\frac{k'}{l-2k'}OPT$ edges from E_{opt}.*

Proof. For any two different values of i from I, the two sets of levels from Observation 7.6.4 are disjoint. Thus if for all $i \in I$, the sets of levels would contain strictly more than $\frac{k'}{l-2k'}OPT$ edges from E_{opt}, then all these levels would contain strictly more than

$$\sum_{m=0}^{\lfloor \frac{l}{k'} - 2 \rfloor} \frac{k'}{l-2k'}OPT = \lfloor \frac{l}{k'} - 1 \rfloor \frac{k'}{l-2k'}OPT \geq (\frac{l-k'}{k'} - 1)\frac{k'}{l-2k'}OPT = OPT$$

edges from E_{opt}, a contradiction. □

Now for each $i \in I$, we use Lemma 7.6.1 to compute optimal solutions to Minimum H-Subgraph Edge Deletion for all l-outerplanar subgraphs. Since for all values of i together, the number of l-outerplanar subgraphs is bounded by O(n), this can be done in O$(n^2 2^{18l^2+9lQ} l^5 Q^2)$ time. We observe that for each i, any H-subgraph of G is present in at least one of the subgraphs of G induced by this i. Therefore, for each i, the union of the optimal solutions for its induced subgraphs is a set of edges that covers all H-subgraphs in G. The algorithm picks the best of these unions as an approximation to the optimal solution. To see that this approximation is at most

$(\frac{s+1}{s})OPT$, consider again an optimal solution E_{opt} for G. We pick the value i that by Observations 7.6.4 and 7.6.5 has not more than $\frac{k'}{l-2k'}OPT$ edges from E_{opt} in intersecting levels of graphs from G^i. For each element $S \in G^i$ we let E_S be the set of edges in E_{opt} in subgraph S. Furthermore, we let E'_S be the edges in an optimal solution of MINIMUM H-SUBGRAPH EDGE DELETION for S. For this choice of i, we thus have a solution of MINIMUM H-SUBGRAPH EDGE DELETION for G of size no larger than the sum of the $|E'_S|$'s. Clearly for each S it holds that $|E'_S| \leq |E_S|$. Moreover, since at most $\frac{k'}{l-2k'}OPT$ edges are counted twice while summing the $|E_S|$'s, we conclude that $\sum_{S\in G^i} |E'_S| \leq \sum_{S\in G^i} |E_S| \leq \frac{k'}{l-2k'}OPT + OPT = \frac{l-k'}{l-2k'}OPT$. Thus in total time O($n^2$)+O($h^2$)+O($n^2 2^{18l^2+9lQ} l^5 Q^2$) we constructed a solution of size at most $\frac{l-k'}{l-2k'}OPT$. By choosing $l = (s+2)k'$, we obtain the $(\frac{s+1}{s})$-approximation to OPT. Since s is strictly positive and we assume that $k > (s+2)k'$, we ensure that $2k' \leq l \leq k$. □

Next, we present an approximation scheme for MINIMUM H-SUBGRAPH EDGE DELETION on planar graph G and pattern H that is either a K_3 or a K_4.

Theorem 7.6.6. *For planar text graph G, pattern H that is either a K_3 or K_4 and any $s > 0$, there is a O($n^2 2^{18l^2} l^5 \binom{3l-1}{h-1} h^2$)-time algorithm for* MINIMUM H-SUBGRAPH EDGE DELETION *on G that finds a solution of size at most $(\frac{s+1}{s})OPT$, where l is a constant depending on s and the outerplanarity index of H.*

Proof. Same as proof of Theorem 7.6.3, with the only difference that Lemma 7.6.2 is used instead of Lemma 7.6.1. □

7.7 Generalization to set $\mathcal{H}$ of patterns

In this final section of the chapter we generalize the results from the previous sections. We show how the algorithms can be adopted to deal with a finite set $\mathcal{H} = \{H_1, \ldots, H_t\},\ t > 1$ of fixed, connected patterns instead of with a single fixed, connected pattern H. Consider such a set of patterns $\mathcal{H} = \{H_1, \ldots, H_t\},\ t > 1$. First we note that if $H_i \in \mathcal{H}$ is a subgraph of $H_j \in \mathcal{H}$, $j \neq i$, then H_j is redundant. Indeed, a set of edges that covers all occurrences of H_i as a subgraph of G will also cover all occurrences of H_j as a subgraph of G. Therefore any graph G that is H_i-free is also H_j-free. For the set of cliques this means that only the smallest clique is significant and hence there is no need to generalize the result from Section 7.5. We thus generalize the results from Section 7.4. To that end, we consider text graph G of bounded maximum degree and a finite set $\mathcal{H}$ of fixed connected patterns such that for each $1 \leq i \neq j \leq t$, H_i is not isomorphic to a subgraph of H_j. By h_i, d_i and k_i we denote respectively the *size*, the *diameter* and the *outerplanarity index* of pattern H_i. The following theorem then generalizes Theorem 7.4.8.

Theorem 7.7.1. *Given a text graph G of bounded maximum degree $\Delta(G)$, a finite set of fixed, connected patterns $\mathcal{H}$ and a nice tree decomposition (T, X) of G of bounded width w, the problem* Minimum H-Subgraph Edge Deletion *on G can be solved in $O(n2^{2w^2+3(w+1)Q}w^5Q^2)$ time, where $Q = \sum_{i=1}^{t} h_i!\binom{\Delta(G)^{d_i+1}}{h_i}$.*

Proof. We repeat the proof of Theorem 7.4.8, taking the given value Q as a new upper bound on the number of considered subgraphs of G. □

Similarly, we generalize Theorem 7.6.3.

Theorem 7.7.2. *For a planar text graph G of bounded degree, finite set of fixed, connected patterns $\mathcal{H}$ and any $s > 0$, there is a $O(n^2 2^{18l^2+9lQ} l^5 Q^2)$-time algorithm for* Minimum H-Subgraph Edge Deletion *that finds a solution of size at most $(\frac{s+1}{s})OPT$, where $Q = \sum_{i=1}^{t} h_i!\binom{\Delta(G)^{d_i+1}}{h_i}$ and l is a constant depending on s and the largest outerplanarity index over all elements from $\mathcal{H}$.*

Proof. The proof is similar to the proof of Theorem 7.6.3, with the difference that $k' = \max_{1 \le i \le t} k_i$ to guarantee that each occurrence of a pattern from $\mathcal{H}$ as a subgraph of G is present in at least one subgraph from $G^i, i \in I$. □

7.8 Conclusions and open questions

We analyzed the problem of excluding a set $\mathcal{H}$ of patterns as subgraphs of a graph G by deleting a (minimum) number of edges from G. By a general result from Courcelle and through a formulation of the problem in Monadic Second Order Logic, we implicitly showed that the decision version of the problem is solvable in linear time on graphs that have bounded treewidth. Subsequently, we presented a constructive algorithm solving the optimization version of the problem in linear time by using dynamic programming on a tree decomposition of the input graph. For this, however, we needed the additional assumption that the input graph has bounded maximum vertex degree, except for the case where $\mathcal{H}$ contains only cliques. Using Baker's layerwise decomposition approach, we created a polynomial time approximation scheme for the problem on planar graphs, using the same additional assumption. It should be noted that the constants in the time complexities of both the linear time algorithm as the PTAS are immense, severely limiting their practical usage. The question remains open whether a combinatorial linear time algorithm and a PTAS can be found that do not require the condition concerning maximum vertex degree in the input graph.

Chapter 8

Summary

In this thesis we consider a set of problems from the areas of computational geometry and algorithmic graph theory. We are driven by an idea of finding a geometric property which provides some insights about the solution of the considered problem. **Part I** of the thesis is devoted to one computational geometry problem. In this area, a common problem is: given a terrain and a moving object together with its speed and other parameters find the optimal trajectory with respect to some objective, for example, time, distance, price and so on, see [2, 35, 90, 95].

It is often assumed that the speed of this moving object is constant; this assumption may significantly simplify all the calculations. The situation is completely different when we assume that the speed of a mover depends on its position in space. We consider a helicopter as an example of such a mover and model it in several ways. In particular, we use the following physical restriction: when a helicopter goes up its maximum velocity goes down as the air density decreases and the power of the rotor drops. Although the decreasing function of the helicopter's maximum velocity in altitude is quite complex and hardly admits a closed analytical form, it is widely accepted to model it by linear or piecewise linear function with a small number of breakpoints, see, e.g. [8].

So, our first natural step is to model the helicopter's velocity as a linear function. We are looking for the fastest trajectory from one point to another, see **Chapter 3**. We use some techniques of calculus of variations to tackle this problem; and we reduce it to the well-known l'Hôpital's light propagation problem [9]. The fastest trajectory appears to be a circle segment. Furthermore, we model the obstacles (terrain) as a continuous rectilinear curve in two-dimensional space and show that the fastest trajectory within this terrain can be found in time, polynomial in the number of obstacles. It is an open question whether it is possible to obtain a polynomial time algorithm for an arbitrary continuous obstacle curve.

We generalize the previous model by making the velocity function piecewise lin-

ear, see **Chapter 4**. It means that within each layer the velocity degradation rate is different. We use the results of the previous chapter and conclude that the fastest trajectory will be a combination of circle segments. This setting leads to a multivariable system of polynomial equations. We use some results from algebraic elimination theory, see, e.g. [33], to reduce this system to one univariable polynomial equation which can be solved by some quick numerical methods like Jenkins-Traub algorithm, see [77]. Some questions stay open. The first question is whether there exists a polynomial time solution to this problem with obstacles in 2D. The second, even more challenging question is whether there are efficient polynomial time approximation schemes for the general helicopter problem with obstacles in 3D.

Parts II and **III** are devoted to various problems involving the treewidth of a graph. The notion of treewidth was introduced by Robertson and Seymour in 1983 [80, 82]. It indicates how much the considered graph is tree-like. Apparently, a lot of hard problems which are easy on trees, are also easy on graphs of small treewidth, see, e.g. [31, 32]. Therefore, the problems of determining the treewidth of a graph or approximating it with a high precision are important for many applications.

We start with planar graphs which do not have bounded treewidth; the question whether or not computing the treewidth of planar graphs is *NP*-hard is still open. But it is known that there is a strong connection between the following parameters of a planar graph: the treewidth, the branchwidth and the largest square-grid minor. There were a number of publications obtaining the upper bounds on the treewidth of a planar graph in terms of the branchwidth and the largest square-grid minor, see, e.g. [52, 56, 83, 89]. The best known result is by Gu and Tamaki [56] which claims that the treewidth is at most 4.5 times the size of the largest square-grid minor. We study a lower bound on this upper bound of the treewidth (see **Chapter 5**) and construct a class of planar graphs where the branchwidth is equal to the largest square-grid minor and where the treewidth is $\frac{3}{2}$ times away from them. This result is followed by Gu and Tamaki [56]; they introduce a class of planar graphs where the treewidth is 2 times away from the largest square-grid minor and is equal to the branchwidth. One of the most interesting questions left is whether there exists a class of planar graphs such that all three parameters are pairwise different.

We continue with the treewidth of general graphs in **Chapter 6**. Two Integer Linear Programming formulations for the problem of finding the treewidth of a general graph are considered. The first formulation is based on the vertex elimination order formulation by Koster and Bodlaender [65]. This formulation can be merged with the flow metric approach by Bornstein and Vempala [24]. Our resulting new ILP cuts some feasible symmetric solutions in the linear relaxation of the original ILP by new inequalities. Furthermore, we introduce a new structural ILP formulation for the treewidth problem based on a drawing of the optimal tree-decomposition on a grid. This new formulation fits within the framework of the local branching technique by Fischetti and Lodi [46]. Therefore, we obtain an exact algorithm for this problem. The computational experiments on different graph classes are left for further research; they may demonstrate the strong and weak points of the proposed ILPs.

Finally, in **Part III**, **Chapter 7**, one application of a treewidth-based approach is introduced. We consider the problem of excluding a subgraph of the given graph by deleting a minimum number of edges from it. This problem is *NP*-hard on general graph instances. We show that by a general result of Courcelle [31, 32], the problem is theoretically solvable in linear time on graphs of bounded treewidth. However, this general approach is not practically useful because of the big constant hidden in an algorithm's running time. Therefore, we present a linear time combinatorial algorithm for a class of graphs with a bounded maximum vertex degree. Moreover, we show that the optimal solution of the problem can be approximated using Baker's type techniques if the input graph is planar, see [6]. The question remains open whether a combinatorial linear time algorithm and a PTAS can be found that do not require the condition concerning maximum vertex degree in the input graph.

Bibliography

[1] P.K. Agarwal, Har-Peled S., Sharir M. and Varadarajan K.R.: *Approximating Shortest Paths on a Convex Polytope in Three Dimensions.* JACM, **44**(4) (1997), pp. 567–584.

[2] P.K. Agarwal, Raghavan P., and Tamaki H.: *Motion Planning for a Steering Constrained Robot through Moderate Obstacles.* In Proc. of the 27th Annual ACM Symposium on Theory of Computing (STOC 1995) (1995), pp. 343–352.

[3] E. Amir: *Efficient Approximation for Triangulation of Minimum Treewidth.* In Proc. of the 17th Conference in Uncertainty in Artificial Intelligence (UAI01), (2001), pp. 7–15.

[4] S. Arnborg, Lagergren J. and Seese D.: *Easy Problems for Tree-Decomposable Graphs.* J. Algorithms, **12**(2) (1991), pp. 308–340.

[5] S. Arora: *Polynomial Time Approximation Schemes for Euclidean Traveling Salesman and Other Geometric Problems.* J. ACM, **45**(5) (1998), pp. 753–782.

[6] B.S. Baker: *Approximation Algorithms for* NP-*Complete Problems on Planar Graphs.* J. ACM, **41**(1) (1994), pp. 153–180.

[7] D.J. Balkcom and Mason M.T.: *Time Optimal Trajectories for Bounded Velocity Differential Drive Vehicles.* I. J. Robotic Res., **21**(3) (2002), pp. 199–218.

[8] *Basic Helicopter Handbook.* US Department of Transportation, Federal Aviation Administration, Advisory Circular 61-13A, (1973).

[9] D.P. Bertsekas: *Dynamic Programming and Optimal Control.* Athena Scientific (1995).

[10] J.T. Betts: *Survey of Numerical Methods for Trajectory Optimization.* J. of Guidance, Control, and Dynamics, **21**(2) (1998), pp. 193–207.

[11] J.T. Betts: *Practical Methods for Optimal Control Using Nonlinear Programming.* SIAM Press, Philadelphia (2001).

[12] Z. Bian and Gu Q.P.: *Computing Branch Decomposition of Large Planar Graphs.* In Proc. of the 7th International Workshop in Experimental Algorithm (WEA 2008), Lecture Notes in Computer Science, **5038**, Springer (2008), pp. 87–100.

[13] D. Bienstock and Monma C.L.: *On the Complexity of Embedding Planar Graphs To Minimize Certain Distance Measures.* Algorithmica, **5**(1) (1990), pp. 93–109.

[14] H.L. Bodlaender: *A Linear-Time Algorithm for Finding Tree-Decompositions of Small Treewidth.* SIAM J. Comput., **25**(6) (1996), pp. 1305–1317.

[15] H.L. Bodlaender: *A Partial k-Arboretum of Graphs with Bounded Treewidth.* Theor. Comput. Sci., **209**(1-2) (1998), pp. 1–45.

[16] H.L. Bodlaender: *Discovering Treewidth.* In Proc. of the 31st Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2005), Lecture Notes in Computer Science, **3381**, Springer (2005), pp. 1–16.

[17] H.L. Bodlaender, Gilbert J.R., Hafsteinsson H. and Kloks T.: *Approximating Treewidth, Pathwidth, Frontsize, and Shortest Elimination Tree.* J. Algorithms, **18**(2) (1995), pp. 238–255.

[18] H.L. Bodlaender, Grigoriev A. and Koster A.M.C.A.: *Treewidth Lower Bounds with Brambles.* Algorithmica, **51**(1) (2008), pp. 81–98.

[19] H.L. Bodlaender and Koster A.M.C.A.: *Safe Separators for Treewidth.* Discrete Mathematics, **306**(3) (2006), pp. 337–350.

[20] H.L. Bodlaender, Koster A.M.C.A. and Eijkhof F. van den: *Pre-processing Rules for Triangulation of Probabilistic Networks.* Computational Intelligence, **21** (2005), pp. 286-305.

[21] T. Bohman: *The Triangle-free Process.* Advances in Mathematics, **221**(5) (2009), pp. 1653–1677.

[22] B. Bollobás: *Extremal Graph Theory.* Handbook of combinatorics (vol. 2), MIT Press, Cambridge, MA (1995).

[23] J.A. Bondy and Murty U.S.R.: *Graph Theory.* Springer Publishing Company, Inc. (2008).

[24] C.F. Bornstein and Vempala S.: *Flow Metrics.* Theor. Comput. Sci., **321**(1) (2004), pp. 13–24.

[25] O.V. Borodin, Glebov A.N., Raspaud A. and Salavatipour M.R.: *Planar Graphs without Cycles of Length from 4 to 7 are 3-Colorable.* J. Comb. Theory, Ser. B, **93**(2) (2005), pp. 303–311.

[26] D. Brügmann, Komusiewicz C. and Moser H.: *On Generating Triangle-Free Graphs.* Electronic Notes in Discrete Mathematics, **32** (2009), pp. 51–58.

[27] J. Canny and Reif J.H.: *New Lower Bound Techniques for Robot Motion Planning Problems.* In Proc. of the 28th Annual IEEE Symposium on Foundations of Computer Science (FOCS 1987), (1987), pp. 49–60.

[28] H. Chen, C. Qiao, F. Zhou and Cheng C.-K.: *Refined Single Trunk Tree: a Rectilinear Steiner Tree Generator for Interconnect Prediction.* In Proc. of 2002 International Workshop on System-Level Interconnect Prediction (SLIP 2002) (2002), pp. 85–89.

[29] H. Chitsaz and LaValle S.M.: *Minimum Wheel-Rotation Paths for Differential Drive Mobile Robots among Piecewise Smooth Obstacles.* In Proc. of the IEEE International Conference on Robotics and Automation (ICRA 2007) (2007), pp. 2718–2723.

[30] H. Chitsaz, LaValle S.M. and Balkcom D.J.: *Minimum Wheel-Rotation Paths for Differential-Drive Mobile Robots.* I. J. Robotics Res., **28**(1) (2009), pp. 66–80.

[31] B. Courcelle: *The Monadic Second-Order Logic of Graphs. I. Recognizable Sets of Finite Graphs.* Inf. and Comput., **85**(1) (1990), pp. 12–75.

[32] B. Courcelle: *The Monadic Second-Order Logic of Graphs III: Tree-decompositions, Minor and Complexity Issues.* ITA, **26** (1992), pp. 257–286.

[33] D. Cox, Little J. and O'Shea D.: *Ideals, Varieties and Algorithms*, Springer-Verlag, New York (1992).

[34] P. Crescenzi, Battista G. D. and Piperno A.: *A Note on Optimal Area Algorithms for Upward Drawings of Binary Trees.* Comput. Geom., **2** (1992), pp. 187–200.

[35] O. Daescu, Mitchell J.S.B., Ntafos S., Palmer J.D. and Yap C.K.: *k-Link Shortest Paths in Weighted Subdivisions.* In Proc. of the 9th International Workshop on Algorithms and Data Structures (WADS 2005), Lecture Notes in Computer Science, **3608**, Springer (2005), pp. 325–327.

[36] O. Daescu, Mitchell J.S.B., Ntafos S., Palmer J.D. and Yap C.K.: *Approximating Minimum-Cost Polygonal Paths of Bounded Number of Links in Weighted Subdivisions.* In Proc. of the 22th ACM Symposium on Computational Geometry (SoCG 2006) (2006), pp. 483–484.

[37] E.D. Demaine and Hajiaghayi M.: *Approximation Schemes for Planar Graph Problems.* Encyclopedia of Algorithms, Springer (2008).

[38] E.D. Demaine and Hajiaghayi M.: *Bidimensionality.* Encyclopedia of Algorithms, Springer (2008).

[39] R. Diestel: *Graph Theory.* Springer-Verlag, Heidelberg (2010).

[40] R. Diestel, Jensen T.R., Gorbunov K.Y. and Thomassen C.: *Highly Connected Sets and the Excluded Grid Theorem.* J. Comb. Theory, Ser. B, **75**(1) (1999), pp. 61–73.

[41] J.E. Dunbar and Frick M.: *The Path Partition Conjecture is True for Claw-free Graphs.* Discrete Mathematics, **307**(11-12) (2007), pp. 1285–1290.

[42] P. Erdös, Kleitman D.J. and Rothschild B.: *Asymptotic Enumeration of K_n-free Graphs.* Atti Dei Convegni Lincei 17, Colloquio Internazionale sulle Teorie Combinatorie, **20** (1976), pp. 19–27.

[43] U. Feige, Hajiaghayi M. and Lee J.R.: *Improved Approximation Algorithms for Minimum Weight Vertex Separators.* SIAM J.Comput., **38**(2) (2008), pp. 629–657.

[44] U. Feige and Lee J.R.: *An Improved Approximation Ratio for the Minimum Linear Arrangement Problem.* Inf. Process. Lett., **101**(1) (2007), pp. 26–29.

[45] G.M. Fichtenholz: *Differential- und Integralrechnung.* B. I–III. Translated from the Russian. VEB Deutscher Verlag der Wissenschaften, Berlin (1986).

[46] M. Fischetti, and Lodi A.: *Local Branching.* Math. Program., **98**(1-3) (2003), pp. 23–47.

[47] F.V. Fomin, Kratsch D, Todinca I., Villanger Y.: *Exact Algorithms for Treewidth and Minimum Fill-In.* SIAM J. Comput., **38**(3) (2008), pp. 1058–1079.

[48] M.R. Garey and Johnson D.S.: *The Rectilinear Steiner Tree Problem in NP Complete.* SIAM Journal of Applied Mathematics, **32** (1977), pp. 826–834.

[49] I.M. Gelfand and Fomin S.V.: *Calculus of Variations.* Dover Publications, Inc., New York (2000).

[50] I.M. Gelfand and Fomin S.V.: *Calculus of Variations.* Gos. Izd. fiz.-mat. lit., Moscow (1961).

[51] *Graph Theory, Combinatorics and Algorithms.* In M.C. Golumbic and Hartman I.B.-A. (Eds.) *Interdisciplinary Applications*, Springer (2005).

[52] A.Grigoriev: *Treewidth and Large Grid Minors in Planar Graphs.* Discrete Mathematics and Theoretical Computer Science, **13**(1) (2011), pp. 13–20.

[53] J.L. Gross and Yellen J.: *Graph Theory and Its Applications.* Second Edition (Discrete Mathematics and Its Applications), Chapman & Hall/CRC (2005).

[54] M. Grötschel, Lovász L. and Schrijver A.: *Geometric Algorithms and Combinatorial Optimization.* Ser. Algorithms and Combinatorics, **2**, Springer (1988).

[55] Q.P. Gu and Tamaki H.: *Optimal Branch-Decomposition of Planar Graphs in $O(n^3)$ Time.* ACM Transactions on Algorithms, **4**(3) (2008), pp. 30:1–30:13.

[56] Q.P. Gu and Tamaki H.: *Improved Bounds on the Planar Branchwidth with Respect to the Largest Grid Minor Size.* In Proc. of the 21st International Symposium on Algorithms and Computation (ISAAC 2010) Part II, Lecture Notes in Computer Science, **6507**, Springer (2010), pp. 85–96.

[57] I.V. Hicks: *Planar Branch Decompositions I: The Ratcatcher.* INFORMS Journal on Computing, **17**(4) (2005), pp. 402–412.

[58] I.V. Hicks: *Planar Branch Decompositions II: The Cycle Method.* INFORMS Journal on Computing, **17**(4) (2005), pp. 413–421.

[59] I.V. Hicks: *Graphs, Branchwidth, and Tangles! Oh my!* Networks, **45**(2) (2005), pp. 55–60.

[60] F. Kammer: *Determining the Smallest k such that G is k-Outerplanar.* In Proc. of the 5th Annual European Symposium on Algorithms (ESA 2007), Lecture Notes in Computer Science, **4698**, Springer (2007), pp. 359–370.

[61] R.M. Karp: *Reducibility Among Combinatorial Problems.* In R.F. Miller and Thatcher J.W. (Eds.) *Complexity of Computer Computations*, Advances in Computing Research, Plenum Press (1972), pp. 85–103.

[62] H.J. Keisler: *Elementary Calculus: An Infinitesimal Approach.* Prindle, Weber, and Schmidt, Boston, MA (1986).

[63] M. Kline: *Calculus: an Intuitive and Physical Approach.* Dover Publications, Inc., New York (1998).

[64] T. Kloks: *Treewidth: Computations and Approximations.* Lecture Notes in Computer Science, **842**, Springer-Verlag, Heidelberg (1994).

[65] A.M.C.A. Koster and Bodlaender H.L.: *Private communication.*

[66] K. Kuratowski: *Sur le Problème des Courbes Gauches en Topologie.* Fund. Math., **15** (1930), pp. 271–283.

[67] C.G. Lekkerkerker and Boland J.C.: *Representation of Finite Graphs by a Set of Intervals on the Real Line.* Fund. Math., **51** (1962), pp. 45–64.

[68] D. Manocha: *Solving Systems of Polynomial Equations.* IEEE Comput. Graph. Appl., **14** (1994), pp. 46–55.

[69] G.J. Minty: *On Maximal Independent Sets of Vertices in Claw-free Graphs.* J. Comb. Theory, Ser. B, **28**(3) (1980), pp. 284–304.

[70] J.S.B. Mitchell and Papadimitriou C.H.: Planning Shortest Paths. In Proc. of the SIAM Conference on Geometric Modeling and Robotics (1985), pp. 1–21.

[71] J.S.B. Mitchell and Sharir M.: *New Results on Shortest Paths in Three Dimentions.* In Proc. of the 20th ACM Symposium on Computational Geometry (SoCG 2004) (2004), pp. 124–133.

[72] J. Palsberg and Naik M.: *ILP-based Resource-aware Compilation.* In A. Jerraya and Wolf W. (Eds) *Multiprocessor Systems-on-Chips*, Elsevier, (2004).

[73] C.H. Papadimitriou: *The Euclidean Traveling Salesman Problem is NP-Complete.* Theor. Comput. Sci., **4**(3) (1977), pp. 237–244.

[74] C.H. Papadimitriou and Steiglitz K.: *Combinatorial Optimization: Algorithms and Complexity.* Prentice-Hall, Inc., Upper Saddle River, NJ (1982).

[75] G. Pokam and Bodin F.: *Energy-Delay Tradeoff Analysis of ILP-based Compilation Techniques on a VLIW Architecture.* INRIA Research Report, **RR-5026**, (2003).

[76] L.S. Pontryagin, Boltyanskii V.G., Gamkrelidze R.V. and Mishchenko E.F.: *The Mathematical Theory of Optimal Processes.* John Wiley & Sons Inc. (1962).

[77] W.H. Press, Teukolsky S.A., Vetterling W.T. and Flannery B.P.: *Numerical Recipes in C. The Art of Scientific Computing.* Cambridge University Press, Cambridge (1992).

[78] S. Rao, and Richa A.W.: *New Approximation Techniques for Some Linear Ordering Problems.* SIAM J. Comput., **34**(2) (2004), pp. 388–404.

[79] D.S. Richeson: *Eulers Gem: The Polyhedron Formula and the Birth of Topology.* Prinston and Oxford, Prinston University Press (2008).

[80] N. Robertson and Seymour P.D.: *Graph Minors. I. Excluding a Forest.* J. Comb. Theory, Ser.B, **35**(1) (1983), pp. 39–61.

[81] N. Robertson and Seymour P.D.: *Graph Minors. III. Planar Tree-width.* J. Comb. Theory, Ser. B, **36**(1) (1984), pp. 49–64.

[82] N. Robertson and Seymour P.D.: *Graph Minors. II. Algorithmic Aspects of Tree-Width.* J. Algorithms, **7**(3) (1986), pp. 309–322.

[83] N. Robertson and Seymour P.D.: *Graph Minors. X. Obstructions to Tree-decomposition.* J. Comb. Theory, Ser. B, **52**(2) (1991), pp. 153–190.

[84] D.J. Rose, Tarjan R.E. and Lueker G.S.: *Algorithmic Aspects of Vertex Eliminination on Graphs.* SIAM J. Comput., **5**(2) (1976), pp. 266–283.

[85] N. Sbihi: *Algorithme de Recherche d'un Stable de Cardinalité Maximum dans un Graphe sans Étoile.* Discrete Mathematics, **29**(1) (1980), pp. 53–76.

[86] P.D. Seymour and Thomas R.: *Graph Searching and a Min-Max Theorem for Tree-Width.* J. Comb. Theory, Ser. B, **58**(1) (1993), pp. 22–33.

[87] P.D. Seymour and Thomas R.: *Call Routing and the Ratcatcher.* Combinatorica, **14**(2) (1994), pp. 217–241.

[88] J.A. Storer and Reif J.H.: *Shortest Paths in the Plane with Polygonal Obstacles.* J. ACM, **41**(5) (1994), pp. 982–1012.

[89] R. Thomas: *Tree-Decompositions of Graphs.*
`http://people.math.gatech.edu/~thomas/SLIDE/CBMS/trdec.pdf`

[90] A. Venkatraman and Bhat S.P.: *Planar Time-Optimal and Length-Optimal Paths under Acceleration Constraints.* In Proc. of Americal Control Conference (2006), pp. 5219–5224.

[91] K. Wagner: *Über eine Eigenshaft der ebenen Complexe.* Math. Ann., **14** (1937), pp. 570–590.

[92] S. Warner: *Modern Algebra.* Prentice-Hall, Inc., (1965).

[93] J.H. Wilkinson: *The Evaluation of the Zeros of Ill-Conditioned Polynomials. Parts i and ii.* Numer. Math., **1**(1), Springer Berlin-Heidelberg (1959), pp. 150–180.

[94] M. Yannakakis: *Edge-Deletion Problems.* SIAM J. Comput., **10**(2) (1981), pp. 297–309.

[95] J.W. Yeol, Ryu Y.S. and Montalvo M.A.: *Shortest Trajectory Planning of Wheeled Mobile Robots with Constraints.* In Proc. of Networking, Sensing and Control, IEEE (2005), pp. 883–888.

Nederlandse Samenvatting

In dit proefschrift staan een aantal vraagstukken uit de gebieden van computationele geometrie en algoritmische grafentheorie centraal. Een belangrijke leidmotief bij de behandeling van deze vraagstukken vormt het zoeken naar en het benutten van geometrische eigenschappen die inzicht bieden in mogelijke oplossingen en oplosmethoden.

Deel I van het proefschrift is gewijd aan de computationele geometrie. Een bekend vraagstuk binnen dit vakgebied is het bepalen van het optimale traject voor een bewegend object met betrekking tot een gegeven grootheid, bijvoorbeeld snelheid, afstand of prijs, zie [2, 35, 90, 95]. Vaak wordt hierbij verondersteld dat de snelheid van het object constant is; een aanname die de benodigde berekeningen vergemakkelijkt. De complexiteit van het probleem neemt aanzienlijk toe wanneer wordt verondersteld dat de maximale snelheid van het object afhangt van zijn positie in de bewegingsruimte. Als voorbeeld van zo'n bewegend object wordt in dit proefschrift een helikopter genomen. Verschillende varianten van het vinden van het optimale traject worden in deze dissertatie vervolgens gemodelleerd. De gebruikte relatie tussen de maximale snelheid en de ruimtelijke positie wordt bepaald door de volgende fysieke eigenschap: wanneer een helikopter omhoog vliegt, neemt de luchtdichtheid af wat resulteert in een afname van de stuwkracht van de rotoren en dientengevolge in een reductie van de snelheid. De daadwerkelijke relatie tussen snelheid en hoogte is redelijk complex en is nauwelijks in gesloten analytische vorm uit te drukken. In de literatuur is het echter gebruikelijk en geaccepteerd om de maximale snelheid van een helikopter te modelleren als een lineaire dalende functie van de hoogte of als een stuksgewijs lineaire functie met een klein aantal breekpunten, zie [8].

Allereerst wordt de snelheid van de helikopter dan ook als een lineaire functie gemodelleerd. **Hoofdstuk 3** beschrijft de situatie waarin de helikopter zich beweegt in een tweedimensionale ruimte met obstakels, die bijvoorbeeld bergen kunnen voorstellen. Een obstakel wordt gemodelleerd als een continue rechtlijnige curve. De vraagstelling in dit hoofdstuk is het bepalen van de snelste route voor de helikopter tussen twee gegeven punten in deze ruimte. Om dit probleem op te lossen worden verschillende technieken uit de variatierekening (onderdeel van de functionaalanalyse) aangewend en wordt het probleem uiteindelijk gereduceerd tot het licht propagatie probleem van l'Hôpital [9]. De route die de helikopter dient te volgen om zo snel mogelijk zijn doel te bereiken blijkt uiteindelijk een cirkelsegment te zijn. Betreffende de complexiteit van het probleem wordt als resultaat gevonden dat het optimale traject bepaald kan worden in tijd, die polynomiaal is in het aantal obstakels. Of er ook voor de situatie waarin obstakels als willekeurige continue curven gemodelleerd worden een algoritme bestaat met een polynomiale doorlooptijd, blijft een open vraag.

In **Hoofdstuk 4** wordt het bovenstaande model gegeneraliseerd door de maximale snelheid van de helikopter als stuksgewijze lineaire functie van de hoogte te modelleren. In dit hoofdstuk wordt de lucht dus gezien als een niet uniform medium dat bestaat uit verschillende lagen. Binnen elke luchtlaag is het tempo waarin de snelheid afneemt met de hoogte verschillend. Mede door gebruik te maken van de resultaten uit het vorige hoofdstuk volgt als resultaat in dit hoofdstuk dat het optimale traject in een tweedimensionale ruimte in deze situatie bestaat uit een combinatie van cirkelsegmenten. De wiskundige beschrijving van de situatie in dit hoofdstuk leidt tot een multivariaat model met polynomiale vergelijkingen. Door gebruik te maken van resultaten uit de eliminatie theorie binnen de algebraïsche meetkunde, zie [33], wordt dit systeem gereduceerd tot een polynomiale vergelijking in één variabele, die met behulp van snelle numerieke methoden, zoals het Jenkings-Traub algoritme, opgelost kan worden, zie [77]. Enkele vragen blijven onopgelost. Eén daarvan is of er een polynomiale tijd algoritme bestaat voor de gegeneraliseerde situatie met obstakels in de tweedimensionale ruimte. Een meer uitdagende open vraag is of er een efficiënt polynomiaal benaderingsschema bestaat voor het gegeneraliseerde helikopter probleem met obstakels in de driedimensionale ruimte.

Deel II en **III** van dit proefschrift zijn gewijd verschillende aspecten van de boombreedte van een graaf. De notie van boombreedte van een graaf is geïntroduceerd door Robertson en Seymour in 1983 [80, 82]. Het vormt een maat voor de gelijkenis die de graaf vertoont met een samenhangende graaf zonder cykels; een zogenaamde boom. Menig NP-volledig graaftheoretisch probleem laat zich in polynomiale tijd oplossen op een boom. Het is een bekend gegeven dat dergelijke problemen ook vaak in polynomiale tijd oplosbaar zijn op grafen die een lage boombreedte hebben, zie bijvoorbeeld [31, 32]. Het bepalen of het nauwkeurig benaderen van de boombreedte van een graaf is dan ook een relevant vraagstuk met vele nuttige toepassingen.

Allereerst wordt in **Hoofdstuk 5** aandacht besteed aan de boombreedte van planaire grafen. De grafen in deze klasse hebben geen begrensde boombreedte. De vraag of het bepalen van de boombreedte van planaire grafen een NP-moeilijk probleem is, staat nog altijd open. Wel is bekend dat er een connectie bestaat tussen de volgende parameters van een planaire graaf: de boombreedte, de zogenaamde 'takbreedte' en de (grootte van de) grootste zogenaamde 'vierkante-grid minor' van de graaf. Er zijn een aantal publicaties waarin bovengrenzen worden verkregen voor de boombreedte van een planaire graaf in termen van de takbreedte en de grootste vierkante-grid minor van de graaf, zie o.a. [52, 56, 83, 89]. Eén van de verkregen resultaten is dat de boombreedte van een planaire graaf hooguit viereneenhalf maal zo groot kan zijn als de grootste vierkante-grid minor in de graaf. In **Hoofdstuk 5** wordt onderzocht in hoeverre deze bovengrens nog naar beneden kan worden bijgesteld. Er wordt een klasse van planaire grafen geconstrueerd waarvoor de takbreedte gelijk is aan de grootste vierkante-grid minor en waarvoor de boombreedte anderhalf keer zo groot is als de takbreedte. Deze ondergrens voor de bovengrens is later verbeterd door Gu en Tamaki [56], die een klasse van planaire grafen construeren waarvoor de boombreedte gelijk is aan de takbreedte en tweemaal zo groot is als de grootste vierkante-grid minor. Een interessante open vraag is of er een klasse van planaire grafen bestaat waarvoor de drie parameters paarsgewijs van elkaar verschillen.

Vervolgens wordt in **Hoofdstuk 6** de focus verlegd naar de studie van boombreedte van grafen in het algemeen. Het boombreedte probleem wordt in dit hoofdstuk op twee manieren als een geheeltallig lineair programmeringsprobleem geformuleerd. De eerste formulering is afgeleid van de 'knoop eliminatie volgorde' formulering zoals die is opgesteld door Koster en Bodlaender [65]. Deze formulering wordt gefuseerd met de 'stroom metriek' aanpak van Bornstein en Vempala [24]. Door het toevoegen van een aantal nieuwe ongelijkheden wordt in onze eerste formulering het toegelaten oplossingsgebied van de relaxatie van de formulering van Koster en Bodlaender gereduceerd. Aansluitend daarop wordt in **Hoofdstuk 6** een tweede geheeltallig lineair programmeringsprobleem voor boombreedte geformuleerd. Deze nieuwe structurele formulering is gebaseerd op een inbedding van een optimale boom decompositie in een grid graaf, die past binnen het raamwerk van een 'lokale vertakkings methode', die bedacht is door Fischetti en Lodi [46]. Gebruikmakend van dit gegeven wordt in dit proefschrift een exact algoritme voor het probleem geconstrueerd. Om meer inzicht te krijgen in de sterke en zwakke punten van beide boombreedte formuleringen zouden in een vervolgonderzoek rekenexperimenten op verschillende klassen van grafen uitgevoerd moeten worden.

Tenslotte wordt in **Deel III** van het proefschrift een toepassing behandeld van het gebruik van boombreedte en boom decomposities in algoritmes. Het vraagstuk wat daarvoor wordt gebruikt is het uitsluiten van een bepaalde subgraaf in een graaf door het verwijderen van een minimum aantal lijnen uit de graaf. Dit probleem is NP-moeilijk voor algemene grafen. Uit een bekend resultaat van Courcelle [31, 32] volgt dat het probleem theoretisch oplosbaar is in lineaire tijd op grafen met begrensde boombreedte. Deze benadering zou echter leiden tot een algoritme met een enorme verborgen constante term in de O()-notatie van de looptijd, en zou derhalve niet praktisch zijn. In **Hoofdstuk 7** wordt voor het beschreven subgraaf eliminatie probleem een combinatorisch algoritme gepresenteerd waarvan de looptijd lineair is, voor grafen waarvan de maximum graad van de knopen begrensd is. Verder wordt in dit hoofdstuk aangetoond dat een optimale oplossing voor het subgraaf eliminatie probleem op planaire grafen kan worden benaderd door gebruik te maken van een techniek, die bedacht is door Baker [6]. Het blijft vooralsnog een open vraag of er ook voor grafen waarvoor de graad van de knopen niet begrensd is een lineaire tijd algoritme en een polynomiaal benaderingsschema voor het subgraaf eliminatie probleem bestaat.

Curriculum Vitae

Natalya Usotskaya was born on October 25, 1983 in Kurgan, Russia. In June 2000, she received her high school diploma from the Specialized Educational Scientific Center of Novosibirsk State University in Russia. In September 2000, she started studying Mathematics with specialty Applied Mathematics and Informatics at Novosibirsk State University. She received her Master's degree in June 2006. From September 2006 till May 2008, Natalya was employed as a PhD-student at Novosiborsk State University and as a teaching assistant at Siberian State University of Telecommunication and Information Sciences. In June 2008, she joined the Quantitative Economics department of Maastricht University as a PhD-student and became a part of the Operations Research group. The results of her research are presented in this thesis.