Combinatorial and Geometric Aspects of Computational Network Construction

Algorithms and Complexity

zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften

von der Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Marcus Krug

aus Erfurt

Tag der mündlichen Prüfung:	21. Dezember 2011
Erster Gutachter:	Frau Prof. Dr. Dorothea Wagner
Zweiter Gutachter:	Herr Prof. Dr. Michael Kaufmann

Acknowledgments

First of all I would like to thank Dorothea Wagner for the opportunity to work in an interesting and multifaceted field of computer science and for the freedom to choose which problems to work on. When I was a student, her lecture on theoretical computer science fascinated and inspired me and has changed my view on the field. I would also like to thank Michael Kaufmann for accepting to review my thesis despite the fact that he was already reviewing two other theses at the time.

I am grateful to all members of Dorothea Wagner's group for the friendly, stimulating and productive atmosphere they provided. Working in Dorothea Wagner's group was a great experience and I always enjoyed my time in the office. I owe special thanks to Reinhard Bauer and Ignaz Rutter for proof-reading large portions of my thesis and for the fun we had during work. Additionally, I would like to thank Reinhard for his friendship and his good-humored nature that would always lift my spirit.

Finally, I would like to thank all the people outside the group who supported me in various ways during the time it took to write this thesis. Specifically, I would like to thank Henning Piezunka for his longlasting friendship, our numerous conversations and his advice. Most of all, however, I would like to thank my wife Susanne for her inspiration, her constant support, her patience and her faith in me.

Deutsche Zusammenfassung

Ein Netzwerk stellt ein System von miteinander verbundenen Einheiten dar, dessen zugrundeliegende Verbindungs-Struktur mathematisch durch einen Graphen beschrieben werden kann. Die Einheiten eines Netzwerkes werden dabei typischerweise als Knoten und die Verbindungen zwischen diesen als Kanten bezeichnet. Netzwerke treten in unterschiedlichen Ausprägungen und Anwendungsgebieten auf – von Infrastruktur-Netzwerken wie dem Strom-Netz über Transistor-Netzwerke im Schaltkreis-Entwurf bis hin zu sozialen Netzwerken wie facebook oder myspace. Häufig erfüllen die Netzwerke darüber hinaus wichtige Funktionen, wie etwa das Trinkwasser-Netzwerk, das Strom-Netzwerk oder das Straßen-Netzwerk. Aufgrund der wichtigen Funktionen, die diese Netzwerke für unser Leben übernehmen, ist ein reibungsloser Betrieb sowie die Effizienz der Netzwerke von entscheidender Bedeutung und sie müssen daher gemäß sorgfältig ausgewählter Kriterien geplant und konstruiert werden.

Netzwerk-Konstruktion bezeichnet einen Prozess, bei dem eine gegebene Menge von Knoten gemäß vorgegebener Kriterien in einem Netzwerk miteinander verbunden werden. Häufig soll die Qualität des resultierenden Netzwerkes bezüglich eines oder mehrerer Qualitätsmaße optimiert werden. Ein Strom- oder Trinkwasser-Netzwerk etwa sollte möglichst ausfallsicher sein, so dass die Beschädigung einer einzelnen Leitung nicht die Versorgung einer größeren Menge von Haushalten betrifft. Ein Transport-Netzwerk wie das Straßen-Netzwerk hingegen sollte schnelle Verbindungen ohne große Umwege zwischen beliebigen Orten bereitstellen. Die zunehmende Komplexität dieser Netzwerke, wie etwa im Schaltkreis-Entwurf, führt dazu, dass viele Konstruktionsprobleme für Netzwerke nur noch mithilfe von Rechnern bewältigt werden können. Bereits 1969 untersuchte daher etwa Scott das sogenannte "Optimal Network Problem" [Sco69] aus algorithmischer Sicht. Bei diesem Problem soll auf einer gegebenen Menge von Knoten in der Ebene ein Netzwerk konstruiert werden, das eben jene Umwege minimiert, indem die Summe der Längen aller kürzesten Wege minimiert werden soll. Diese Größe zur Beschreibung der Effizienz eines Transport-Netzwerkes wird auch als *Routing-Kosten* bezeichnet.

Algorithmische Netzwerk-Konstruktions-Probleme treten in vielfältiger Form und in einer Vielzahl von Anwendungsgebieten auf. So reicht das Spektrum der algorithmischen Netzwerk-Konstruktions-Probleme von grundlegenden, rein kombinatorischen Problemen wie dem Aufzählen und zufälligen Erzeugen von Graphen bis hin zu komplexen, multi-kriteriellen Optimierungsproblemen aus dem Bereich der Generalisierung von Landkarten. Die vorliegende Arbeit spannt einen Bogen von elementaren hin zu komplexen und von rein kombinatorischen hin zu geometrischen Netzwerk-Konstruktions-Problemen sowie den damit verbundenen Netzwerk-Einbettungs-Problemen.

Im ersten Teil der Arbeit beschäftige ich mich mit *kombinatorischen* Netzwerk-Konstruktions-Problemen mit steigender Anzahl von Einschränkungen und Optimierungskriterien. Am Anfang der Betrachtung stehen dabei Probleme ohne Optimierungsfunktion mit rein strukturellen Einschränkungen. Anschließend untersuche ich klassische Optimierungsprobleme mit einer Optimierungsfunktion sowie multi-kriterielle Optimierungsprobleme mit zwei Optimierungsfunktionen.

Im zweiten Teil der Arbeit wende ich mich *geometrischen* Netzwerk-Konstruktions-Problemen zu, die sich dadurch auszeichnen, dass die Knoten des zu konstruierenden Netzwerks in die euklidische Ebene eingebettet werden müssen. In diesem Kontext ergeben sich eine Reihe spezieller Einschränkungen und Optimierungskriterien. Im Folgenden sind die betrachteten Problemstellungen sowie meine Lösungsansätze kurz skizziert.

Aufzählungs-Algorithmen und Graph-Generatoren für degenerierte Graphen Beim Aufzählen und zufälligen Erzeugen von Graphen sollen zu einer gegebenen Menge von Eigenschaften *alle* Netzwerke oder *ein zufälliges* Netzwerk mit den gewünschten Eigenschaften konstruiert werden. Da diese Problemstellungen keine weitere Optimierung der konstruierten Netzwerke hinsichtlich eines Qualitätsmaßes fordern, zählen sie zu den elementarsten Netzwerk-Konstruktionsproblemen. Während Aufzählungs-Algorithmen etwa bei erschöpfender Suche zur Anwendung kommen, werden Graph-Generatoren zur Simulation sowie zur experimentellen Analyse von Algorithmen benötigt.

Ein Graph ist k-degeneriert, wenn jeder induzierte Teilgraph einen Knoten mit Grad höchstens k enthält. Zu den degenerierten Graphen zählen Bäume, da jeder Teilbaum einen Knoten mit Grad höchstens 1 enthält, sowie planare Graphen, da jeder Teilgraph eines planaren Graphen einen Knoten mit Grad höchstens 5 enthält. Da selbst für reguläre Graphen, eine weitere spezielle Klasse von degenerierten Graphen, nicht bekannt ist, ob diese gleichverteilt unter Berücksichtigung von Isomorphie erzeugt werden können, betrachte ich das Problem Vernachlässigung von Isomorphie. Dies entspricht einem gängigen Vorgehen, bei dem die Knoten eines Graphen je mit einem eindeutigen Label annotiert werden, um diese voneinander unterscheiden zu können. Diese Label werden jedoch nur während der Erzeugung verwendet und anschließend vernachlässigt. Ich betrachte dabei eine neue Art von Labeling, die es mir ermöglicht derart annotierte degenerierte Graphen effizient aufzuzählen sowie gleichverteilt zu erzeugen. Ein Vorteil dieser neuen Methode ist, dass die dadurch induzierte Verteilung auf den nicht annotierten Graphen näher an der Gleichverteilung der degenerierten Graphen unter Berücksichtigung von Isomorphie ist als die durch ein klassisches Labeling induzierte Verteilung.

Optimale Erweiterung von Baum-Netzwerken bezüglich Routing-Kosten Die Optimierung der Netzwerke spielt bei vielen Netzwerk-Konstruktions-Problemen eine zentrale Rolle. So sollen etwa typischerweise die Konstruktions-Kosten des Netzwerks minimiert und die Funktionalität des Netzwerks maximiert werden. Aufgrund ihrer vergleichsweise geringen Konstruktions-Kosten spielen Bäume im Bereich der Netzwerk-Konstruktion eine wichtige Rolle. Als klassisches Netzwerk-Optimierungsproblem mit lediglich einer Optimierungsfunktion betrachte ich die Optimierung gewichteter Routing-Kosten bei der Erweiterung von Netzwerken. Bei dieser Variante der Routing-Kosten sind die kürzesten Wege gemäß der erwarteten Anzahl von Nutzern, die diese Wege benutzen werden, gewichtet. Zunächst untersuche ich dabei das Problem, zwei unzusammenhängende Bäume durch Erweiterung mit einer einzelnen Kante zu verbinden, so dass die Routing-Kosten minimiert werden. Diese Problemstellung entsteht bei einem Zusammenschluss von Netzwerken sowie nach dem Ausfall einer Kante. Hierzu betrachte ich sowohl das Erweiterungs- wie auch das Reparatur-Problem für verschiedene Distanzfunktionen zwischen den Knoten der Bäume. Anschließend beschäftige ich mich mit dem Problem, einen gegebenen, gerichteten Baum mit einer Abkürzung zu erweitern, die

die Routing-Kosten minimiert, um die Performanz des Netzwerkes zu steigern. Während sich beide Probleme durch Aufzählen aller möglichen Kanten trivialerweise in quadratischer Zeit lösen lassen, zeige ich wie man diese Probleme durch Reduktion auf geometrische Konzepte deutlich effizienter und in einigen untersuchten Varianten sogar optimal lösen kann.

Das Dichte-Maximierungs-Problem in Graphen Viele Netzwerk-Konstruktionsprobleme, die durch realitätsnahe Problemstellungen motiviert sind, erfordern die gleichzeitige Optimierung von mehreren Optimierungsfunktionen. Da Optimalität in diesem Fall nicht eindeutig definiert ist, existieren verschiedene Konzepte im Umgang mit mehreren Optimierungsfunktionen wie etwa Pareto-Optimalität sowie die Zusammenfassung der Optimierungsfunktionen durch gewichtete Summen oder gewichtetes Minimum bzw. Maximum. Ich betrachte Optimierungsprobleme mit zwei Optimierungsfunktionen, bei denen eine *Kosten*-Funktion minimiert und eine *Nutzen*-Funktion maximiert werden soll. In diesem Fall kann man Optimalität bezüglich des Quotienten von Nutzen und Kosten definieren, was zur Folge hat, dass jedes bezüglich des Quotienten optimale Ergebnis auch Pareto-Optimal ist, darüber hinaus aber zusätzlich das beste "Kosten-Nutzen"-Verhältnis aufweist. Dieses Verhältnis entspricht damit dem "Return-On-Investment" einer Netzwerk-Konstruktion. Ich untersuche die Komplexität sowie effiziente Algorithmen für die Maximierung des Kosten-Nutzen-Quotienten für verschiedene Varianten des Problems. Während einige der untersuchten Varianten NP-schwer oder sogar nicht bis auf einen konstanten Faktor approximiert werden können, falls $\mathcal{P}$ ungleich $\mathcal{NP}$ ist, zeige ich für einige Varianten effiziente Approximations-Algorithmen sowie Fixed-Parameter-Algorithmen auf.

Orthogeodätische Einbettungen von Graphen Geometrische Netzwerk-Konstruktionsprobleme sind dadurch charakterisiert, dass sie zusätzlich zur Konstruktion des Netzwerkes eine Einbettung desselben in eine Teilmenge der euklidischen Ebene erfordern, bei der Knoten auf Punkte und Kanten auf Jordan-Kurven abgebildet werden. Häufig wird dabei aus Gründen der Komplexität der Problemstellung die Konstruktion des Netzwerkes von der Einbettung des Netzwerkes in die Ebene getrennt, wie etwa beim Schaltkreis-Entwurf.

Während geradlinige Einbettungsfragen bereits vielfach untersucht wurden, betrachte ich *orthogeodätische Einbettungen*, eine spezielle neue Variante von orthogonalen Einbettungen, bei denen Kanten auf *monotone* kürzeste Ketten von orthogonalen Segmenten abgebildet werden. Damit bilden die entstehenden Kurven das kanonische Gegenstück zu geradlinigen Segmenten auf einem orthogonalen Gitter. Der resultierende Zeichenstil liefert daher funktionale und übersichtliche Einbettungen.

Zunächst untersuche ich verschiedene Varianten des kreuzungsfreien, orthogeodätischen Einbettungsproblems aus algorithmischer Sicht. Dabei zeige ich, dass man das Einbettungsproblem effizient lösen kann, sofern man lediglich fordert, dass die Knoten und Kanten auf einem ganzzahligen Gitter quadratischer Größe liegen sollen. Anschließend beweise ich, dass das Problem NP-schwer ist, wenn die Knoten des Netzwerkes nur auf eine kleine, beschränkte Menge von Punkten abgebildet werden dürfen. Für den Spezialfall, dass das Netzwerk ein Kreis ist, kann ich hingegen eine einfache Charakterisierung sowie einen effizienten Algorithmus angeben, der entscheidet, ob eine gegebene Menge von Punkten eine Einbettung des Kreises zulässt und der eine solche Einbettung auch berechnet, sofern sie existiert. Weiter veranschauliche ich, dass das Problem auch dann noch NP-schwer ist, wenn man die Positionen

der Knoten fest vorgibt und fordert, dass die Kanten auf dem Gitter eingebettet werden sollen. Ohne diese Forderung jedoch gebe ich einen effizienten, zertifizierenden Algorithmus an, der entweder eine Einbettung der Kanten berechnet oder einen leicht zu überprüfenden Beweis dafür liefert, dass eine solche Einbettung nicht existiert.

Anschließend nähere ich mich dem orthogeodätischen Einbettungsproblem für Bäume aus kombinatorischer Sicht. Hierzu untersuche ich für verschiedene Klassen von Bäumen und für verschiedene orthogeodätische Einbettungsvarianten die minimale Anzahl von Punkten mit der Eigenschaft, dass *alle* Punktmengen dieser Größe die Einbettung *aller* betrachteten Bäume bezüglich der gegebenen Einbettungsvariante zulassen. Hierbei berücksichtige ich Bäume und sogenannte Raupen-Graphen mit Maximalgrad 3 und 4 sowie Einbettungsvarianten mit ein oder zwei Knicken pro Kante. Darüber hinaus betrachte ich die Problemstellung sowohl unter Berücksichtigung wie auch unter Vernachlässigung von Planarität. Für alle betrachteten Varianten gebe ich jeweils ich eine obere Schranke für die gesuchte minimale Größe der Punktmenge an.

Häufig muss die Einbettung der Netzwerke unter bestimmten Einschränkungen erfolgen, etwa wenn die Knoten des Netzwerkes nicht beliebig auf die Punkte abgebildet werden können. Dieses Szenario lässt sich dadurch modellieren, dass man davon ausgeht, dass die Punkte gefärbt sind. Ich betrachte eine Variante des Problems, bei welcher die Punkte rot beziehungsweise blau gefärbt sind und bei welcher benachbarte Knoten des Netzwerkes auf unterschiedlich gefärbte Punkte abgebildet werden müssen. Für dieses Problem beweise ich, dass jede Punktmenge mit etwa gleicher Anzahl von roten und blauen Punkten eine orthogeodätische Einbettung eines Pfades selber Größe zulässt, sofern jede horizontale und jede vertikale Gerade höchstens einen Punkt enthält. Die Einschränkung auf Punktmengen mit dieser Eigenschaft ist dadurch motiviert, dass es im allgemeinen beliebig große Punktmengen gibt, die eine orthogeodätische Einbettung bestimmter Bäume nicht zulassen, etwa eine Menge von Punkten auf einer einzigen horizontalen Linie. Darüber hinaus erläutere ich einen effizienten Algorithmus, der eine solche orthogeodätische Einbettung berechnet. Fordert man hingegen, dass der Pfad auf dem Gitter eingebettet werden muss, wird das Problem erstaunlicherweise NP-schwer. Motiviert durch dieses Resultat untersuche ich daher die Länge eines längsten Pfades. Ich beweise, dass es gefärbte Punktmengen gibt, für die der längste Pfad höchstens etwa die Hälfte der Punkte enthalten kann und gebe einen effizienten Approximations-Algorithmus an, der einen Pfad mit mindestens einem Drittel der Punkte berechnet.

Generalisierung von geometrischen Graphen Als letzten Themen-Komplex betrachte ich das Problem der Generalisierung von geometrischen Graphen. Dieses Problem ist motiviert durch die steigende Menge zu visualisierender Daten und die daraus resultierende Notwendigkeit der Daten-Reduktion. Obgleich sich Zeichnungen von riesigen Graphen effizient berechnen lassen [KCH03, HJ05], reichen die Ressourcen selbst modernster Medien nicht aus, um Netzwerke mit mehreren Millionen von Knoten darzustellen. Bisherige Ansätze zum Umgang mit dieser Tatsache lösen das Problem nicht zufriedenstellend. So verzerren Fish-Eye Visualisierungen die Zeichenebene zulasten des Gesamteindrucks und Algorithmen zur Bündelung von Kanten führen häufig zu sehr stark veränderten Darstellungen.

Das Problem der Generalisierung von geometrischen Graphen besteht darin, zu einer gegebenen Zeichnung eines Netzwerkes ein kleineres Netzwerk zu konstruieren, das dem

gegebenen sowohl in geometrischer als auch in graph-theoretischer Hinsicht ähnlich ist. Bereits 1995 hatte Saalfeld [Saa95] gefordert, das Generalisierungsproblem von Landkarten aus graph-theoretischer-theoretischer Sicht zu betrachten und in einem rigiden mathematischen Rahmen zu analysieren. Ich modelliere dieses Problem erstmals in seiner vollen Breite in einem mathematischen Modell, indem ich verschiedene Artefakte, die bei der Visualisierung von großen Netzwerken in einem Medium mit beschränkter Auflösung entstehen, zur Formulierung von Optimierungsproblemen nutze, die darauf abzielen, ebendiese Artefakte zu vermeiden. Ich zeige, dass die entstehenden multi-kriteriellen Problemstellungen im Allgemeinen NP-schwer sind, und gebe darüber hinaus effiziente Approximations-Algorithmen sowie Heuristiken an, die in einer experimentellen Studie bereits sehr gute Ergebnisse lieferten.

Contents

Chapter 1

Introduction

Networks have played an important role in the development of modern culture and technology. While social networks have helped develop the complex societies of today's world, infrastructure networks have been the basis for many far-reaching achievements in the history of human kind. Even in ancient times, infrastructure networks such as waterways and road networks have played a crucial role for merchandise and information exchange over different cultures and peoples and have, thus, provided the basis for the deployment of novel ideas and technologies. Irrigation networks have made it possible to detach farmers' fates from the unpredictabilities of nature and, thus, have permitted families to abandon nomadism to settle down and found cities. As these cities grew larger and larger it was again an infrastructure network, the drainage system, that helped coping with the increasing number of inhabitants. Much later, the industrial revolution drew much of its momentum from the introduction of the railway system and the 20th century has implemented such infrastructure networks as power supply systems, gas pipelines and long-distance heating systems. But even today's world and much of our well-being still depends on rapidly growing networks such as water and power supply systems, gas pipelines, road networks and airways. Today we are witnessing some of the largest networks installed by humans such as the Internet and the transistor networks on modern CPUs with up to two billion nodes, and these networks, too, are playing a revolutionary role for economies, cultures and even politics.

What all of these networks have in common is that they are designed for a special purpose or functionality and that they are vital to a large number of people. Further, they are rather expensive to construct and to maintain and the efficiency of these networks crucially depends on an appropriate design. Thus, they must be planned and constructed with special care.

Today we are facing a large number of network construction problems involving *tangible* networks such as power and water supply systems, road networks and communication networks. The rapid growth of existing networks, for instance, as a consequence of rapidly growing populations in developing countries or the increasing mobility, but also the desire to abandon fossil sources of energy in favor of regenerative sources, involve the construction of large new parts of infrastructure networks. These networks must be carefully planned and optimized since they must be highly efficient and their construction involves huge costs. For instance, as energy from regenerative sources is seldom produced where it is consumed, we need to establish new infrastructure networks to distribute this energy adequately. However, the economical and ecological consequences of such networks may have a severe long-term impact on the surroundings of their erection and, thus, these networks should be designed thoroughly.

A large number of network construction problems resulting, for instance, during the operation of a network, involve the constructing a *virtual* sub-network in a given network, rather than constructing a tangible network. As typical examples, consider transportation problems evolving in logistics, such as the problem of distributing goods to customers by a transportation company or the problem of setting up a communication scheme among a large number of participants in a wireless communication network. Whether a company is capable of constructing these virtual distribution networks in an adequate way or not typically has a large impact on its competitiveness and, therefore, on its economic success.

Given that networks are almost ubiquitous and play such a crucial role in today's world, it is not surprising that the study of the optimal design of networks has received considerable attention in various areas of research. To name just a few examples, Jha et al. study design issues of road network construction [JSJK06] and Sarte studies design principles of sustainable green infrastructure networks [Sar10]. Further, network design issues play an important role in the area of chip design as illustrated in the introductory textbook by Carballo [Car08]. Apart from that, sensor networks spur increasing interest in the algorithmic construction of communication networks among sensors [Kn09]. Finally, it is interesting to note that even social networks such as facebook and myspace depend on a careful design specifying the rules according to which the network may evolve and, thus, be constructed by the people engaging in them. These issues are studied, for instance, in [How10].

What is considered an optimal design, however, strongly depends on the purpose of the network. Various qualities such as reliability and performance must be quantified in order to assess how well a given network design is suited for its specific task. While reliability is typically quantified by the number of network components that may fail without discontinuing the network's operation, the performance of a network can be quantified, for instance, by its routing cost, defined as the total length of all shortest paths in the network, or by its dilation, a factor quantifying the maximum detour for routing in the network as compared to some optimal reference path. On the other hand, the construction of a network often involves construction costs of some kind. While this is immediately obvious for infrastructure networks involving hardware, this equally applies to virtual networks, such as communication networks, where each additional link may introduce an additional communication overhead that may result in unwanted latency.

As a consequence of the rapid growth of these networks, it is becoming increasingly important to solve the design problems for networks with the aid of computer programs. In fact, optimal network design problems are among the earliest and most extensively studied problems in various areas of computer science. Already in 1926 Borůvka provided an algorithm for the problem of constructing a minimum weight spanning tree of a network as a means to construct an efficient electricity network for the electric power company in Moravia [Bor26]. According to Shrijver [Sch05], the transportation problem, defined as the problem of supplying a given amount of goods to a set of customers with given demands such that the resulting costs are minimized, was considered by Tolstoǐ from a computational point of view as early as 1930. Further, the traveling salesperson problem, one of the most well-known and widely studied problem in computer science, which is to find a shortest tour vising a given number of sites, can be traced back to a manual for traveling salesmen formulated in 1832, but Menger [DS98] seems to be the first to consider this problem, which he called "Das Boten-Problem" from a mathematical point of view almost a century later in 1930.

However, the interest in these problems has not faded over time. In the 1960s and 1970s network construction problems became a hot topic from a computational point of view with the increasing accessibility of computers. In the early 1960s Quandt [Qua60] discussed models for the network construction problems and quantitative measures for optimal network construction. Later, Werner [Wer68] studied the relationship between spatial network design issues and the construction and operation of networks and Scott [Sco69] formulated the *optimal network problem*, which is to find a network interconnecting a given set of sites in the plane without crossings such that the routing cost of the network is minimized. In the 1970s Boyce et al. [BFW73] studied branch-and-bound algorithms for solving the optimal network problem, Johnson et al. [JLK78] studied the complexity of the network design problem and Dionne and Florian [DF79] studied exact and approximate algorithms for optimal network design.

Network construction problems involving integrated circuit design have become increasingly important during the 1980s with the development of very large scale integration production techniques allowing hundreds of thousands of transistors per chip. This development has fueled research in graph drawing, which is closely related to integrated circuit layout. Leiserson [Lei80], Valiant [Val81] and Kramer and van Leeuwen [KvL84], for instance, have studied graph drawing problems with applications in very large scale integration circuit layout, such as the problem of routing wires on the circuit board as well as the problem of minimizing the area of orthogonal layouts.

But even today, many network construction problems constitute a multifaceted and active area of research, such as the construction of communication infrastructure backbones in wireless sensor networks [GK11, PM11, DWW$^+$11] and the traveling salesperson problem [ABCC06, RGGO11, KM11]. Further, *network theory* has evolved as an independent science [Lew09, New10] involving graph theory, the study of the evolution of random networks as well as network analysis [Sco00, McC07] and network visualization [BETT99].

As we have seen from the previous examples, network construction problems have various applications and appear in various areas of computer science. In this thesis we adopt a unified view on a wide range of problems that are related to the construction of networks from a computational point of view. A *computational network construction problem* is a problem whose underlying task is to construct a *graph*, that is, an abstract network, subject to a given set of *hard constraints* and such that a given set of *optimization goals* are met. Depending on the nature of the constraints, the optimization goals and the type of the network that is to be constructed, we distinguish several types of computational network construction problems. For instance, we distinguish between *combinatorial* network construction problems, where the task is to compute a virtual representation of the network as a graph—as in the construction of a wireless communication network—and *geometric* network construction problems, where the task is to compute a geometric network—as in the case of integrated circuit design and the design of road networks. Often the construction of geometric networks is split into a combinatorial network construction problem with the aim of computing the functionality of the network and the construction of the actual layout. That is, in the layout phase we are already given a graph representing the combinatorial structure of the network to be constructed as an input and the task is merely to construct a layout of this graph subject to various constraints and optimization goals. As an example, consider integrated circuit *layout*, a sub-problem of integrated circuit design that is considered independently from circuit design. We refer to this kind of network construction problem as *network embedding problem*. All these problems can be cast as a computational network construction problem by choosing constraints and optimization goals in an adequate way. For instance, we may view the network embedding problem as the special case of a geometric network construction problem in which the topology of the network is part of the hard constraints of the problem.

Thesis Outline

In this thesis, we study a small collection of computational network construction problems from various areas of computer science. We consider these problems from both a computational and a combinatorial point of view and we present results concerning their complexity as well as algorithmic solutions. We start by considering combinatorial network construction problems with increasing complexity of the network's underlying model and optimization goals. Then we turn our attention to network embedding problems and, finally, we study a geometric network construction problem. In the following we present a short overview of the contents of this dissertation.

Chapter 2. Preliminaries
We briefly introduce the main concepts and definitions appearing throughout thesis. We start by reviewing some basic definitions from graph theory as well as geometry and graph drawing. Then we turn to basic concepts underlying the theory of NP-completeness and polynomial-time many-to-one reductions needed for the presented complexity results. As some of the problems considered in this thesis are computationally hard, we then repeat basic notions from the theory of fixed-parameter tractability and approximation algorithms providing the basis for the presented approaches to the computationally hard problems.

Chapter 3. Enumerating and Generating Well-Ordered Degenerate Graphs
The problems of enumerating graphs and generating random graphs can be considered as the most basic combinatorial network construction problems since these problems do not ask for a specific graph but *every* or *a random* graph from a given class of graphs. Enumeration algorithms are used, for instance, for exhaustive search in brute-force-algorithms and graph generators are used, among others, in simulations and experimental analysis of algorithms. To avoid bias, graph generators should produce each graph of a given class of graphs uniformly, that is, with equal probability. Since it is sometimes hard to devise uniform generators for unlabeled graphs, it is a common technique to resort to generating labeled graphs uniformly.

We study the problem of enumerating and generating k-degenerate graphs. While there does not seem to be a straight-forward way of generating ordinary k-degenerate graphs uniformly at random—either labeled or unlabeled—we show that we can uniformly generate well-ordered k-degenerate graphs by introducing a new vertex labeling. This way we generate each unlabeled k-degenerate with positive probability ruling out some, but not all isomorphic copies. We show that the distribution of the generators induced on the unlabeled k-degenerate graphs is closer to the uniform distribution as compared to the classical labeling, which makes this approach preferable to the classical labeling approach. We present efficient algorithms for the enumeration and the random generation of labeled k-degenerate graphs within this new labeling scheme.

Chapter 4. Optimal Routing Cost Tree Augmentation
We consider two network augmentation problems on trees asking for the insertion of a single edge such that resulting network is optimized with respect to routing cost. These problems evolve when two networks must be re-combined, for instance, after a single link has broken

down, or if the performance of the network is to be increased. While both problems can be solved trivially by exhaustive search over all pairs of endpoints in quadratic time, we present faster algorithms with optimal or near-optimal running times exploiting geometry.

First, we consider the problem of making a network consisting of two trees connected by introducing an additional edge such that the routing cost of the resulting network is minimized. We study this problem for various distance measures on and between the trees. We show that the problem can be efficiently solved by reducing it to the problem of computing a Voronoi diagram in the plane or in a graph, respectively. Moreover, we show that the quadratic-time exhaustive algorithm is worst-case optimal if we do not require the distance measure to be metric.

Second, we consider the problem of introducing a shortcut into a weighted directed tree-network such the weighted routing cost is minimized. We show that this problem can be efficiently solved by reducing it to the computation of the upper envelope of an associated set of piecewise linear functions. We show that the latter task can be solved in linear time for the resulting piecewise linear functions.

Chapter 5. The Density Maximization Problem in Graphs

Many network design problems involve maximizing the performance of the network while minimizing the construction cost for building and maintaining the network. We study the density maximization problem in graphs, which, given a graph with edge lengths and edge weights, asks for a subgraph whose length is minimized while its weight is maximized. We tackle this problem by maximizing the ratio of the total weight over the total length, called the density, as an optimization goal. Thus, we compute a Pareto-optimal solution whose weight to length ratio is maximized. By considering the length of the network as costs and the weight as profit, we can re-cast this problem in terms of the optimization of a network's return-on-investment.

We study this problem for different classes of graphs and under various constraints. First, we assume that we are given an upper bound on the total length and a lower bound on the total weight, that is, a limited budget and a target profit. We show that the problem is NP-hard and we provide pseudo-polynomial fixed-parameter tractable algorithms that can be applied to a large class of graphs. Further, we present an FPT-algorithm for the special case that we wish to find a path with maximum density with respect to a structural parameterization of the problem describing how tree-like a graph is. Then we show that the problem admits a fully polynomial-time approximation scheme if the constraint on the length may be violated at the cost of an additional penalty term on the weight.

Second, we study the problem of interconnecting a given set of so-called Steiner vertices such that the density of the resulting network is maximized. We provide NP-hardness and inapproximability results for this case and we present a fixed-parameter tractable algorithm for computing a maximum density path where the parameter is the length of the path. Further, we show that the problem of computing an arbitrary maximum density subgraph containing a given set of Steiner vertices in a planar graph is fixed-parameter tractable when parameterized by the number of vertices of the sought subgraph, whereas we show that this problem is W[1]-hard for general graphs.

Chapter 6. Orthogeodesic Embedding of Planar Graphs
We introduce and study the orthogeodesic drawing style for embedding planar graphs. Since orthogeodesic chains are shortest-possible orthogonal chains they can be considered to be the counterpart of straight lines on the orthogonal grid. In terms of network construction, these chains are optimal connections between their endpoints with respect to construction cost. We consider the problem of embedding a given graph into a subset of the Euclidean plane. First, we consider the orthogeodesic embedding problem without any constraints and show that it can be efficiently solved by reducing it to the problem of computing an orthogonal embedding with at most one bend per edge. Next, we consider the embedding problem for the case when we are given a small set of points in the plane to which the vertices of the graph must be mapped. We show that this problem is NP-hard and we provide a simple characterization of the point sets admitting an orthogeodesic embedding of a cycle as well as an efficient algorithm for computing such an embedding. Finally, we consider the problem for the case, when the mapping between the vertices of the graph and the points is given. While this problem turns out to be NP-hard even for matchings if the embedding must be on the grid, we show that we can efficiently compute such an embedding without this restriction. To this end, we present a certifying algorithm that either computes an embedding or an intelligible proof that no such embedding exists.

Chapter 7. Orthogeodesic Embeddings of Trees
A set of points admitting an embedding of all graphs from a given class of graphs for a specific drawing style is called *universal* for this class of graphs and this drawing style. We consider the combinatorial problem of determining the size of the smallest point set whose points are neither horizontally nor vertically aligned such that *every* point set of is this size is universal for *all* trees with respect to the orthogeodesic drawing style. We consider planar and non-planar orthogeodesic drawing styles with two-bend and one-bend orthogonal chains, respectively, and we provide upper bounds for the size of universal point sets for various classes of trees including trees with maximum degree 3 and 4 as well as caterpillars.

Chapter 8. Bicolored Hamiltonian Orthogeodesic Paths
In the colored point-set embeddability problem we are given a graph and a colored set of points and the problem is to determine whether the graph can be embedded on the colored point set such that adjacent vertices of the graph are mapped to points with different colors. This models a situation in which the embedding of the vertices of the network is underlying a set of addition constraints. We consider a special variant of this problem where the graph is a path and the point set is colored with only two colors. We show that the problem of deciding whether a path can be embedded on the grid is NP-hard and we provide an efficient algorithm if the path needs not be on the grid. Additionally, we provide an efficient algorithm approximating the length of a longest path on the grid up to a factor of 1/3 and we prove that there are point sets such that the longest path contains at most slightly more than half of the points.

Chapter 9. Generalization of Geometric Graphs
When visualizing or constructing geometric graphs, we must usually avoid placing objects too close to each other. This holds equally true if we wish to visualize a graph's vertices and edges on a display media with limited resolution and if we wish to place transistors and route wires on a circuit board. If the vertices of a graph are mapped too close to each other

they are said to clutter. Clutter avoidance is an important problem arising, for instance, in computational cartography and network visualization when zooming out of the drawing, that is, when generalizing the drawing. This motivates the problem of generalizing geometric graphs. This problem is to construct a small graph that is similar with respect to both geometry and structure to the original graph and that can be visualized without clutter on the given display media. We formalize the problem by distinguishing several types of clutter and formulating the avoidance of these types of clutter as optimization problems. We show that these problems are NP-hard, and we therefore devise efficient approximation algorithms and easy-to-implement heuristics. We implemented the heuristics and showcase the results.

Chapter 2

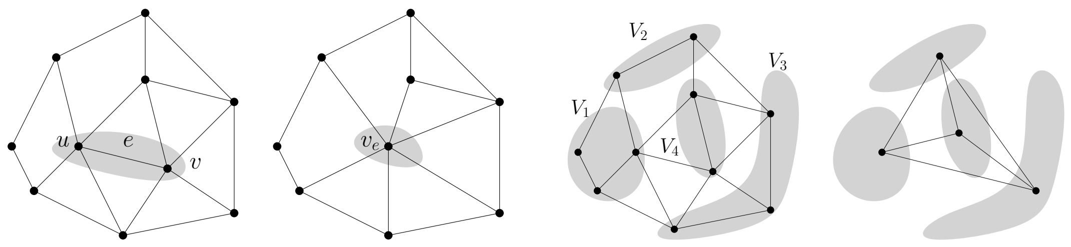
Preliminaries

In this section we introduce basic concepts and definitions from geometry, graph theory, complexity theory and approximation algorithms used throughout the thesis. We assume that the reader is familiar with basic concepts from computer science and algorithmics, especially the Bachmann–Landau notation describing the asymptotic behavior of functions. An introduction to basic algorithmic techniques can be found in the textbook *Introduction to Algorithms* by Cormen et al. [CLRS09].

2.1 Networks and Graphs

The term *network* refers to any set of entities that are associated with each other such that the resulting structure can be described formally as a graph. Thus, a network is a concrete manifestation of a graph, while the latter is an abstract mathematical object. Formally, a *graph* is a tuple $G = (V, E)$ consisting of a set $V = \{v_1, \dots, v_n\}$ of *vertices* and a set $E = \{e_1, \dots, e_m\}$ of *edges* such that E is a binary relation on V. A *geometric graph* is a graph whose vertices are associated with a given set of points in the Euclidean plane $\mathbb{R}^2$ in a one-to-one correspondence. Usually, we identify the vertices of a geometric graph with their associated points. Given a graph $G = (V, E)$, we denote the set of vertices of G by $V(G) := V$ and the set of edges of G by $E(G) := E$, respectively. Throughout the thesis we consider only *finite* graphs, that is, both V and E are finite. The edges of a graph can be either *directed* or *undirected* depending on whether the relation defined by the edge-set is symmetric or not. A directed edge is a tuple $e = (u, v)$ such that $u, v \in V$. We say that e is directed from u to v and we call u the *source* and v the *target* of e, respectively. A *directed graph* is a graph whose edges are directed. An undirected edge is an unordered set $\{u, v\}$ such that $u, v \in V$. An *undirected graph* is a graph whose edges are undirected. The vertices of a directed or undirected edge e are called the *endpoints of* e. If it is clear from the context, whether the graph G is directed or undirected, we will sometimes abbreviatingly write uv instead of $\{u, v\}$ and (u, v). An edge uv such that $u = v$ is called a *loop*. A graph is called *simple* if it contains no loops. A *multi-graph* is a graph whose set of edges is a multi-set, that is, a multi-graph can have multiple edges between two distinct vertices. Two vertices u and v of a graph G are called *adjacent* if there is a directed or undirected edge between u and v. A vertex u that is adjacent to another vertex v is called a *neighbor of* v. The set of neighbors of v is denoted by $N(v)$. A vertex v and an edge e are *incident* if v is an endpoint of e. The degree of v, denoted by $\mathrm{d}(v)$, is equal to the number edges that are incident to v.

A *path* in G is a sequence of vertices $P = (u_1, \dots, u_k)$ such that $k \geq 1$ and such that G contains the edges $u_i u_{i+1}$ for all $1 \leq i \leq k - 1$. The *length* of P is equal to the number



(a) Contraction of a single edge (b) Branch sets and minor

Figure 2.1: Contraction, branch sets and minors. (a) Graph G/e resulting from contracting the edge uv. (b) Graph $G/\mathcal{C}$ with $\mathcal{C} = \{V_1 \dots, V_4\}$.

of edges on P. A path is called *simple*, if the vertices on P are distinct. A *cycle* is a path $C = (u_1, \dots, u_k)$ such that $u_1 = u_k$. If the vertices $u_1, u_2, \dots, u_{k-1}$ are distinct, then C is called *simple*. A *connected graph* is a graph such that each pair of vertices is connected by a path.

Let $G = (V, E)$ be a graph and let $G' = (V', E')$ be a graph with $V' \subseteq V$ and $E' \subseteq E$. Then G' is called a *subgraph of* G, denoted by $G' \subseteq G$. We say that G contains G' if G' is a subgraph of G. If, additionally, E' contains all edges $uv \in E$ such that $u, v \in V'$, then G' is called an *induced subgraph*, that is, a subgraph that is induced by the set of vertices $V' \subseteq V$, denoted by $G[V']$.

Let $G = (V, E)$ and $G' = (V', E')$ be two graphs and let $v \in V$ be a vertex. Then the union of G and G', denoted by $G \cup G'$ is defined as the graph $H = (V \cup V', E \cup E')$. The graph $G[V \setminus v]$ is also denoted by $G - v$. If $U \subseteq V$, we write $G - U$ for the graph $G[V \setminus U]$. Similarly, if $e \in E$ is an edge of G, we denote the graph resulting from G by removing e by $G - e$ and we extend this notation to sets of edges $F \subseteq E$ by writing $G - F$ for the graph we obtain from G by removing the edges in F.

A *tree* is a simple connected graph that does not contain any cycle. A *leaf* of a tree is a vertex whose degree is equal to one. Any vertex of a tree that is not a leaf is an *internal vertex*. Any two vertices in a tree T are connected by a unique path in T. A tree $T = (V, E)$ may contain a special vertex $r \in V$, called *root*. Then T is called a *rooted tree* and we say that T is rooted at r. A root induces a partial order on the vertices of T. If T is rooted in r, we say that $u \in V$ is *below* $v \in V$ (respectively v is *above* u) if v lies on the unique path connecting r to u. The *height* of a rooted tree is the maximum length of a simple path between r and some other vertex of T. A *caterpillar* C is a special tree with the property that the subgraph induced by the internal vertices of C is path. A *forest* is a finite union of trees.

Let $G = (V, E)$ and $G' = (V', E')$ be two graphs and let $\varphi\colon V \to V'$ be a bijection mapping vertices in G to vertices in G'. Then φ is a *graph isomorphism*, or *isomorphism*, for short if there is an edge uv in G if and only if there is an edge $\varphi(u)\varphi(v)$ in G'. If $G = G'$, then an isomorphism between G and G' is called an automorphism. The set of the permutations of the vertices of a graph together with the composition form a group, the *symmetric group of the graph*, in which the set of automorphisms is a subgroup, called the *automorphism group*.

Let $G = (V, E)$ be a graph and let $e = uv$ be an edge in G. By G/e we denote the graph we obtain by *contracting* the edge e into a new vertex $v_e \notin V$ that is adjacent to all neighbors of u and v as illustrated in Figure 2.1a. Formally, G/e is a graph whose vertex set is given by $V' := (V \setminus \{u,v\}) \cup \{v_e\}$ and whose edge set is given by

$$E' := E(G - \{u,v\}) \cup \{v_e w \mid w \in (N(u) \cup N(v)) \setminus \{u,v\}\} .$$

Let $\mathcal{C} = \{V_1, \dots, V_k\}$ be a partition of V such that the induced graphs $G[V_i]$ are connected for all $1 \leq i \leq k$. Then we denote the graph we obtain by contracting all edges in the induced subgraphs $G[V_i]$ by $G/\mathcal{C}$ as illustrated in Figure 2.1b. The sets $V_1, \dots, V_k$ are called *branch sets*. If G is a subgraph of a graph H, then the graph graph $G/\mathcal{C}$ is called a *minor* of H. A family $\mathcal{F}$ of graphs is *closed under taking minors*, or *minor-closed*, if each minor of a graph in $\mathcal{F}$ is also contained in $\mathcal{F}$. As an example, consider the class of forests. Every subgraph F' of a forest F is a forest and by contracting edges in F' we do not create cycles. Therefore, every minor of a forest F is a forest and, thus, the class of forests is closed under taking minors. If $\mathcal{F}$ is a family of graphs that is closed under taking minors, then according to the Robertson-Seymour Theorem [RS04], there is a *finite* set of graphs $\mathcal{Z}$ such that none of the graphs in $\mathcal{Z}$ is contained in $\mathcal{F}$ and such that $\mathcal{Z}$ does not contain any minor of a graph in $\mathcal{Z}$, that is, all minors of graphs in $\mathcal{Z}$ are in fact contained in $\mathcal{F}$. Due to this property $\mathcal{F}$ can also be characterized in terms of the graphs in $\mathcal{Z}$. These graphs are called *forbidden minors* and $\mathcal{F}$ is called *$\mathcal{Z}$-minor-free*. As an example, consider the class of planar graphs. According to Kuratowski's Theorem [Kur30] these graphs can be characterized as the class of graphs that do not have a complete graph on five vertices or a complete bipartite graph on two sets of three vertices each as a minor.

Let $G = (V, E)$ be a graph. A *tree decomposition* of G is a pair $(\mathcal{X}, \mathcal{T})$ such that $\mathcal{X} = \{X_i \mid i \in I\}$ is a collection of subsets of V which are called *bags* and $\mathcal{T} = (I, E_{\mathcal{T}})$ is a tree with the following properties.

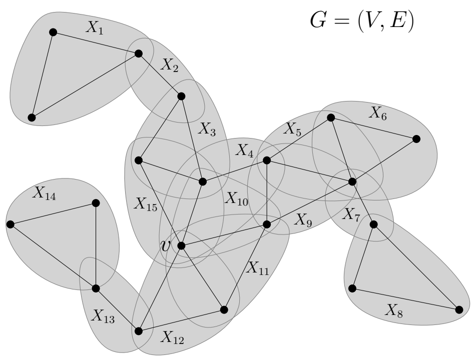
(i) The union of all bags $\bigcup_{i \in I} X_i$ is equal to V.

(ii) For all edges $e \in E$ there is an index $i \in I$ such that $e \subseteq X_i$.

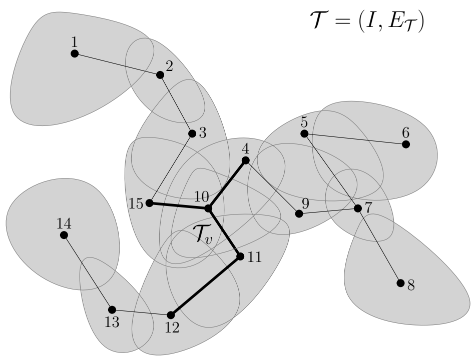
(iii) For all vertices $v \in V$, the tree induced by the set of nodes $\mathcal{X}_v = \{i \in I \mid v \in X_i\}$ induces a connected subtree of $\mathcal{T}$.

We will refer to the elements in I as nodes—as opposed to vertices in the original graph. The *treewidth* of a tree decomposition equals $\max_{i \in I} |X_i| - 1$, that is, the maximum number of vertices over all bags minus one. The treewidth of a graph $G = (V, E)$ is equal to the minimum treewidth of a tree decomposition of G. The treewidth of a tree is equal to one. An exemplary tree decomposition of a graph is illustrated in Figure 2.2. Tree decompositions are often used in combination with dynamic programming on the decomposition tree. An introduction to treewidth and its algorithmic applications is given by Bodlaender and Koster in [BK08].

Quite frequently, the edges of a graph are assumed to have a specific *length*, given by a map $\ell \colon E \to \mathbb{R}$. Then the length of a path in the graph is defined as the sum of the lengths of the individual edges on the path. The length of a shortest path between two vertices u and v of a graph, denoted by $\mathrm{d}_G(u, v)$, defines a metric on the vertices of the graph. We refer to this metric as the distance of u and v in G. For convenience, we sometimes assume that the length of an edge is equal to 1 if no specific length is specified. The *routing cost* of a graph,



(a) Graph and bags



(b) Bags and decomposition tree

Figure 2.2: Tree decomposition $(\mathcal{X}, \mathcal{T})$ with $\mathcal{X} = \{X_1, \dots, X_{15}\}$ and $\mathcal{T} = (I, E_\mathcal{T})$ where $I = \{1, \dots, 15\}$ of a graph $G = (V, E)$ with tree-width two. The tree $\mathcal{T}_v$ induced by $v \in V$ in the decomposition tree is marked boldly.

denoted by rc(G), is defined as the sum of the distances over all pairs of vertices, that is,

$$\operatorname{rc}(G) = \sum_{(u,v)\in V\times V} \operatorname{d}_G(u, v) \ .$$

If we are additionally given a *weight* function $w\colon V \to \mathbb{R}$ on the vertices of G, then we define the *weighted routing cost* as

$$\operatorname{rc}(G) = \sum_{(u,v)\in V\times V} w(u) \cdot w(v) \cdot \operatorname{d}_G(u, v) \ .$$

The definition of the weighted routing cost reflects a situation in which heavy vertices can be considered to be central vertices in charge of distributing traffic. Whereas there is much traffic among heavy vertices, there is little traffic among less heavy vertices.

While the routing cost is a measure for the efficiency of a network, the *construction cost*, defined as the sum of the lengths of all edges, is a measure for how expensive it is to realize such a network. In addition to a length, the edge of a graph are sometimes assumed to have a specific weight $w\colon E \to \mathbb{R}$.

2.2 Geometry and Graph Drawing

We denote the natural numbers by $\mathbb{N}$, the integers by $\mathbb{Z}$ and the real numbers of $\mathbb{R}$; the Euclidean plane is denoted by $\mathbb{R}^2$. For a point $p = (p_x, p_y) \in \mathbb{R}^2$ in the Euclidean plane, we write $x(p) := p_x$ and $y(p) := p_y$, respectively. A point $p = (i, i)$ with integer coordinates $i, j \in \mathbb{Z}$ is called a *grid point*. The *integer grid*, or simply the *grid* for short, is the subset of $\mathbb{R}^2$ consisting of all horizontal and vertical lines defined by the grid points in $\mathbb{Z}^2$. A point is said to be on the grid if it is a grid point and a curve is said to be on the grid if it is a subset of the integer grid. The *bounding box* of a set of points P, denoted by $\mathcal{B}(P)$ is defined as the smallest axis-parallel rectangle containing all points of P.

The straight-line segment between two points p and q in $\mathbb{R}^2$ is denoted by $\overline{pq}$. Unless stated otherwise, the *distance* between two points in the Euclidean plane refers to the L_2-metric

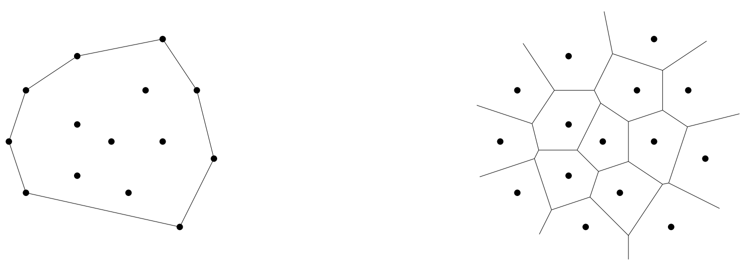

(a) Convex hull (b) Voronoi diagram

Figure 2.3: A plane point set, its convex hull (a) and its Voronoi-diagram (b).

or Euclidean metric. We say that a point $p \in \mathbb{R}^2$ is below (above) $q \in \mathbb{R}^2$ if $y(p) < y(q)$ ($y(q) < y(p)$). Similarly, we say that p is left of q (right of q) if $x(p) < x(q)$ ($x(q) < x(p)$).

A *polygonal chain* consists of a sequence $c = (p_1, \dots, p_n)$ of points in $\mathbb{R}^2$ such that p_i and p_{i+1} are connected by the straight-line segment $\overline{p_i p_{i+1}}$ for all $1 \leq i \leq n-1$. A polygonal chain is called *monotone* if the orthogonal projections of the points $p_1, \dots, p_n$ onto the straight line ℓ defined by its endpoints p_1 and p_n appear in the same order as the points appear on c. The *length* of an orthogonal chain is equal to the sum of the lengths of its straight-line segments. A polygonal chain whose endpoints coincide, that is $p_1 = p_n$, is called a *polygon*. The points on the polygonal chain defining a polygon are also called vertices of the polygon. A polygon whose vertices $p_1, \dots p_{n-1}$ are distinct and whose straight-line segments do not intersect, except in their endpoints, is called *simple*. A simple polygon subdivides the Euclidean plane into two connected regions, a bounded region defining the *inside* of the polygon and an unbounded region defining the *outside* of the polygon.

An *orthogonal chain* is a polygonal chain consisting only of horizontal and vertical segments. An orthogonal chain whose length is equal to the L_1-distance between its endpoints is called an *orthogeodesic chain*. Note that this implies that an orthogeodesic chain is monotone. The points $p_2, \dots, p_{n-1}$ on a polygonal chain are also called *bends*, the points p_1 and p_n are also called endpoints of c. An *L-shaped orthogeodesic chain* is an orthogeodesic chain consisting of exactly one horizontal and one vertical straight-line segment. A *horizontal orthogeodesic chain* is an orthogeodesic chain consisting of two horizontal straight-line segments and one horizontal straight-line segments. Similarly, a *vertical orthogeodesic chain* consists of two vertical and one horizontal straight-line segment. Note that a horizontal (vertical) orthogeodesic chain is uniquely determined by its endpoints and the x-coordinate (y-coordinate) of its vertical (horizontal) straight-line segment.

The *convex hull* of a point set $P \subseteq \mathbb{R}^2$ is the unique simple polygon with vertices in P that contains all straight-line segments between pairs points of points in P in its interior or on its boundary. A set of points and its convex hull is illustrated in Figure 2.3a

The *Voronoi diagram* of a set of points $P \subseteq \mathbb{R}^2$ is a subdivision of the plane into regions R_p for $p \in P$ such that R_p is the locus of points that are closer to p than to any other point $q \in P \setminus \{p\}$ as illustrated in Figure 2.3b. The region R_p is also called the *Voronoi-cell* of p.

A *drawing* of a graph $G = (V, E)$ is a mapping $\gamma \colon V \cup E \to \mathbb{R}^2$ that maps G to a subset of the Euclidean plane $\mathbb{R}^2$ such that every vertex V is mapped to a distinct single point $\gamma(v)$

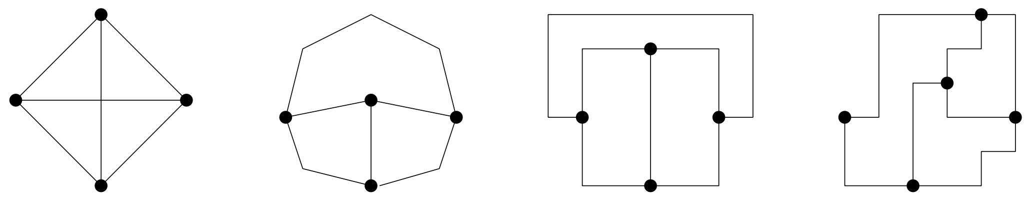


Figure 2.4: Illustration of various graph-drawing styles.

and every edge uv is mapped to an open plane curve $\gamma(uv)$ between its endpoints whose length is finite. That is, γ *embeds* G into $\mathbb{R}^2$. We use the terms drawing and embedding synonymously. Throughout this thesis, we identify u and $\gamma(u)$ for all $u \in V$ as well as e and $\gamma(e)$ for all $e \in E$. A drawing of a graph partitions the plane into a set of connected regions, called *faces*, with exactly one unbounded face, called the *outer face*. A drawing of G that maps V to a given restricted finite set of points P is called a *point-set embedding of G on P*. A graph G is called *planar* if it admits a drawing that does not contain any crossings between the edges. If, additionally there is a drawing of G such that all vertices of G are incident to the outer face, then G is called *outerplanar*.

Depending on the shape of the curve $\gamma(uv)$ to which an edge uv is mapped we distinguish various *drawing styles*. A *straight-line drawing* is a drawing in which each edge is mapped to a straight-line segment as illustrated in Figure 2.4a. Since straight-line segments are uniquely determined by their endpoints and it is sometimes convenient to have more freedom for the placement of the edges, we also consider *polyline drawings*, in which edges are mapped to polygonal chains, that is, edges are allowed to have bends as illustrated in Figure 2.4b. If the polygonal chains are orthogonal, then the drawing is called an *orthogonal drawing* as illustrated in Figure 2.4c. Further, if the polygonal chains are orthogeodesic, then the drawing is called an *orthogeodesic drawing* as illustrated in Figure 2.4d. Graph drawings are discussed in detail in the textbook by Di Battista et al. [BETT99] and in the book by Kaufmann and Wagner [KW01].

2.3 Complexity

In this thesis we study the computational complexity of several problems related to network construction. Naturally, we would like to devise algorithms that solve these problems with as few operations as possible for any given input. Since larger input sizes will typically result in longer running times, we would therefore like to devise algorithms whose asymptotic running time can be bounded by a slowly growing function in the input size. Problems that can be solved within a polynomial number of operations for some polynomial in the input size of the algorithm are commonly considered to be efficiently solvable or tractable, whereas problems requiring a super-polynomial number of operations are not considered to be efficiently solvable. While even algorithms with polynomial running times may be intractable in practice, this classification is well-defined from a theoretical point of view and is therefore widely adopted.

The class of problems that is solvable in polynomial time is denoted by $\mathcal{P}$. However, there is a large number of practically relevant problems for which no efficient algorithms with polynomial running times have yet been found. In order to classify these problems with respect to their complexity, an extensive theory has evolved over time. One of the earliest and most important complexity classes apart from $\mathcal{P}$ is the class $\mathcal{NP}$. This class contains the problems that can be solved by a non-deterministic algorithm in polynomial time. Intuitively, we can think of a non-deterministic algorithm as an algorithm that first guesses a solution to a given problem instance and then verifies this solution. Its running time is said to be polynomial if both the guessing and the verification requires at most a polynomial number of operations in the input size. Thus, the complexity class $\mathcal{NP}$ can equivalently be characterized as the class of problems admitting a small certificate that can be verified in polynomial time. While it is clear that $\mathcal{P} \subseteq \mathcal{NP}$, it is a major open problem in computer science, whether $\mathcal{P} = \mathcal{NP}$ or $\mathcal{P} \neq \mathcal{NP}$. However, many computer scientists believe that $\mathcal{P} = \mathcal{NP}$ is rather unlikely.

Informally, a problem is said to be *NP-hard* if it is at least as hard as any other problem in $\mathcal{NP}$ and it is said to be *NP-complete* if it is contained in $\mathcal{NP}$. Formally, the theory of NP-completeness is built upon the theory of formal languages, Turing machines and polynomial-time reductions between problems. We merely provide a short overview here; for a detailed discussion of the theory of NP-completeness we refer the reader to the textbooks by Garey and Johnson [GJ79] or Cormen et al. [CLRS09]. An *abstract problem* Q is a binary relation between a set of problem instances $\mathcal{I}$ and a set of problem solutions $\mathcal{S}$, that is, Q associates solutions to problem instances. We distinguish between *optimization problems* where the goal is to find a solution that is optimal with respect to some optimization goal and *decision problems* where we are merely to decide whether a given instance has a solution or not. Although many realistic problems are optimization problems by nature, the theory of NP-completeness is built on decision problems. However, each optimization problem can be associated with a decision problem in a straight-forward way and, in some sense, the optimization problems are not harder than their associated decision problems.

While an abstract problem is a binary relation that associates a set of problem instances $\mathcal{I}$ with a set of solutions $\mathcal{S}$, a *decision problem* can be thought of as a function that maps problem instances $\mathcal{I}$ to either *yes* or *no*, or equivalently, to 1 and 0, respectively. That is, each decision problem can be associated with a tuple $(\mathcal{I}, \mathcal{Y})$ such that $\mathcal{I}$ is a set of problem instances and $\mathcal{Y} \subseteq \mathcal{I}$ is the set of *yes*-instances, that is, the set of instances that have a solution with respect to the decision problem Q. Without going into more detail, we briefly mention that decision problems can be related to formal languages by encoding problem instances over a finite alphabet Σ. That is, the *yes*-instances can be associated with a formal language $L \subseteq \Sigma^*$ such that the decision problem is transformed into the problem of recognizing words from L. The *size* of a problem instance is usually measured by the length of the string over Σ corresponding to a problem instance. As long as we use finite alphabets and reasonable encoding schemes, the notion of size is well-defined in that the length of different encodings do not differ too much.

In order to formalize the notion of a problem Q being at least as hard as another problem R, we reduce R to Q, that is, we show that any algorithm deciding Q can be used to decide R after applying some kind of transformation to the problem instances. Formally, a decision problem R is said to be *polynomial-time reducible* to a decision problem Q if there is an algorithm $\mathcal{A}$ that transforms an instance x of R into an instance $\mathcal{A}(x)$ of Q such that the running time of $\mathcal{A}$ is bounded by a polynomial in the size of x and such that $\mathcal{A}(x)$ is a *yes*-instance of Q if and only if x is a *yes*-instance of R. In order to show that a decision problem Q is NP-hard, we must therefore show that an NP-hard problem R is polynomial-time reducible to Q. In order to show that Q is NP-complete, we must show that it is NP-hard and that it is contained in NP, that is, that there is a non-deterministic algorithm with polynomial running time that decides Q. Thus, whenever an NP-hard problem R is polynomial-time reducible to a problem Q, then any polynomial-time algorithm for Q can be used to devise a polynomial time algorithm for R by first transforming the instance x of R using $\mathcal{A}$ and then applying the polynomial-time algorithm on the transformed instance. That is, in some sense Q is at least as hard to solve as R.

Some decision problems, such as the following Partition problem, involve numbers and must be treated with special care.

Problem Partition

Instance: A set of objects $A := \{a_1, \dots, a_n\}$ and a weight $w\colon A \to \mathbb{Z}$.

Solution: A partition $A = A_1 \uplus A_2$.

Goal: Is there a partition $A = A_1 \uplus A_2$ such that $\sum_{a \in A_1} w(a) = \sum_{a \in A_2} w(a)$?

While no algorithm solving the Partition problem is known whose running time is bounded by a polynomial in the input length where each number k must be encoded by at most $\mathcal{O}(\log k)$ digits, there are algorithms whose running time is bounded by a polynomial in the input length and the largest input number. Algorithms for decision problems involving numbers whose running time is bounded by a polynomial in the input size and the largest number are called *pseudo-polynomial*. However, there are problems involving numbers that do not admit a pseudo-polynomial-time algorithm unless $P = NP$. A problem Q involving numbers is called *strongly NP-hard* if there is polynomial p such that a restricted variant of the problem Q_p is NP-hard, where the problem Q_p is obtained from Q by restricting the largest input number of each instance to be bounded by $p(n)$ where n denotes the size of the instance. A problem is said to be *strongly NP-complete* if it is strongly NP-hard and contained in $\mathcal{NP}$. Clearly, strongly NP-hard problems do not admit pseudo-polynomial-time algorithms unless $\mathcal{P} = \mathcal{NP}$. For instance, the following problem is a classical strongly NP-complete problem [GJ79].

Problem 3-Partition

Instance: A set of objects $A := \{a_1, \dots, a_{3m}\}$ and a weight $w\colon A \to \mathbb{Z}$.

Solution: A partition $A = A_1 \uplus A_2 \uplus \cdots \uplus A_m$.

Goal: Is there a partition $A = A_1 \uplus A_2 \uplus \cdots \uplus A_m$ such that $\sum_{a \in A_i} w(a) = B$ for all $1 \leq i \leq m$?

A detailed discussion of the theory of NP-completeness as well as a large number of NP-complete problems can be found in the textbook by Garey and Johnson [GJ79].

2.4 Fixed-Parameter Tractability

While the theory of NP-completeness is a powerful tool for assessing the complexity status of computational problems in terms of the hardest instances of these problems, it does not take into account that some problem instances may be easier to handle than others. Despite the fact that some computational problem may contain very hard instances, it may well be the case that these instances do not—or merely seldom—occur in practical applications. The theory of fixed-parameter tractability seizes this idea by parameterizing the input instances and devising algorithms whose running time is polynomial in the input size, but is allowed to have an additional super-polynomial factor depending on the parameter. If the parameter of an instance is small, then fixed-parameter tractable algorithms can be efficient for practical problems.

A *parameterization* is a function κ that maps problem instances to natural numbers. If the problem instances of a decision problem are graphs, then the maximum degree of the graph is a valid parameterization, for instance. A *parameterized problem* is a tuple (Q, κ) consisting of an decision problem Q and a parameterization κ. The parameterized problem (Q, κ) is called *fixed-parameter tractable (FPT)* if each instance x of Q can be solved in time $\mathcal{O}(f(\kappa(x)) \cdot |x|^c)$ where $|x|$ denotes the size of the input x, c is a constant not depending on the input size and the parameter and f is a computable function. We will equivalently say that the problem Q is fixed-parameter tractable with respect to the parameter κ. An algorithm for a parameterized problem whose running time is $\mathcal{O}(f(\kappa(x)) \cdot |x|^c)$ is called an *FPT-algorithm* and the complexity class *FPT* contains all parameterized problems that are fixed-parameter tractable.

However, not all parameterized problems do admit FPT-algorithms. Thus, a theory of the hardness of parameterized problems has evolved that is analogous to the theory of NP-completeness, albeit, more complex. Similar to the theory of NP-completeness, parameterized problems are related to each other via reductions. A parameterized problem (Q, κ) is *FPT-reducible* to the parameterized problem (Q', κ') if there is an FPT-algorithm $\mathcal{A}$ with respect to κ that transforms each instance x of Q into a new instance $\mathcal{A}(x)$ of Q' such that x is a *yes*-instance of Q if and only if $\mathcal{A}(x)$ is a *yes*-instance of Q' and such that $\kappa'(\mathcal{A}(x))$ is bounded by a computable function in $\kappa(x)$. While there are various hierarchies of complexity classes for parameterized problems, we only briefly mention the classes W[1] and W[2] that appear in this thesis. These classes are the bottommost complexity classes of a hierarchy $W[1] \subseteq W[2] \subseteq \cdots$ of complexity classes that are defined by means of a family of logic decision problems. We omit the details here and merely mention that a parameterized problem (Q, κ) is $W[t]$-hard under FPT-reductions, if every problem in $W[t]$ is FPT-reducible to (Q, κ). That is, in order to show that a parameterized problem is $W[t]$-hard, it suffices to provide an FPT-reduction from a $W[t]$-hard parameterized problem (Q', κ') to (Q, κ). For more details we refer the reader to the textbook by Flum and Grohe [FG06] as well as to the textbook by Niedermeier [Nie06].

2.5 Approximation Algorithms

While fixed-parameter tractable algorithms can be used to compute optimal solutions for problem instances for which the parameter is small, they do not help solving problem instances for which the parameter is large. Another way of approaching NP-hard optimization problems is to try to compute approximate solutions instead of optimal solutions. Formally, an optimization problem is four-tuple $Q = (\mathcal{I}, \mathrm{sol}, m, \mathrm{type})$ where sol associates each problem instance $x \in \mathcal{I}$ with a set $\mathrm{sol}(x)$ of possible solutions such that $y \in \mathrm{sol}(x)$ can be verified in polynomial time. Further, m associates each tuple x and $y \in \mathrm{sol}(x)$ with a number $m(x, y)$ and the type $\in \{\min, \max\}$ denotes the type of the optimization problem, which is either a minimization problem or a maximization problem. The goal of an optimization problem is to find, for a given problem instance x, a solution $y \in \mathrm{sol}(x)$ such that the measure $m(x, y)$ is optimal with respect to type, that is $m(x, y) = \mathrm{type}\{m(x, y') \mid y' \in \mathrm{sol}(x)\}$. The measure $m(x, y)$ is the called the optimum of x, denoted by $\mathrm{opt}(x)$. Given an optimization problem Q, an instance $x \in \mathcal{I}$ and a solution $y \in \mathrm{sol}(x)$, the *performance ratio* or *approximation ratio* of y with respect to x is defined as

$$R(x, y) = \max\left\{\frac{m(x, y)}{\mathrm{opt}(x)}, \frac{\mathrm{opt}(x)}{m(x, y)}\right\} .$$

An algorithm $\mathcal{A}$ that computes for each problem instance $x \in \mathcal{I}$ a solution $\mathcal{A}(x) \in \mathrm{sol}(x)$ such that

$$R(x, \mathcal{A}(x)) \leq r$$

is called an *r-approximation algorithm*. The class of optimization problems that have an r-approximation algorithm for a fixed r is denoted by APX. An optimization problem Q belongs to the class PTAS if there is a family of algorithms containing for each $\varepsilon > 0$ an algorithm $\mathcal{A}_\varepsilon$ that is an $(1 + \varepsilon)$-approximation algorithm for Q. The family of algorithms $\mathcal{A}_\varepsilon$ is called a *polynomial time approximation scheme (PTAS)*. If, additionally, the running time of the algorithm $\mathcal{A}_\varepsilon$ of a PTAS is polynomial in the input size and $1/\varepsilon$, then the family of algorithms is called a *full polynomial-time approximation scheme (FPTAS)*. Clearly, PTAS is contained in APX. However, some optimization problems are not contained in PTAS, unless $\mathcal{P} = \mathcal{NP}$. There are various techniques for showing that an optimization problem is unlikely to be in PTAS by means of approximation-preserving reductions. An overview of approximation-preserving reductions is given by Crescenzi [Cre97]. A more detailed discussion of approximation algorithms can be found in the textbook by Ausiello et al. [ACG$^+$02].

Chapter 3

Enumerating and Generating Well-Ordered Degenerate Graphs

The first network construction problem we consider in this chapter is the problem of enumerating and generating k-degenerate and strongly k-degenerate graphs. A k-degenerate graph is a graph in which every induced subgraph has a vertex with degree at most k and a strongly k-degenerate graph is a k-degenerate graph whose minimum degree is k. Network construction problems of this kind are among the most basic network construction problems since they do not require the construction of an optimal network but instead ask for the construction of either every or an randomly sampled network from a specific class of networks. However, at least as far as generators are concerned, it is desirable to generate the objects uniformly at random, that is, with equal probability, in order to avoid a bias on the resulting distribution of the networks.

The class of k-degenerate graphs includes, among others, planar graphs and it plays an interesting role in the theory of fixed parameter tractability since some otherwise $W[2]$-hard domination problems become fixed-parameter tractable for k-degenerate graphs. Algorithms for enumerating and generating k-degenerate graphs have applications in exhaustive brute-force algorithms and experimental analysis of algorithms, respectively. Additionally, these generators can be used to generate graphs with given core hierarchy. We introduce a novel labeling scheme for the vertices of a k-degenerate graph and call the resulting labeled graphs *well-ordered* k-degenerate graphs.

While it is not clear how to generate k-degenerate graphs uniformly at random either unlabeled or labeled, we present efficient algorithms for the problem of enumerating and generating *well-ordered* k-degenerate graphs uniformly at random. By generating well-ordered k-degenerate graphs we generate at least one labeled copy of each *unlabeled* k-degenerate graph and we filter out some but not all isomorphies compared to the classical labeled approach. Additionally, we introduce the class of strongly k-degenerate graphs, that is, k-degenerate graphs with minimum degree k, which are a natural generalization of k-regular graphs and we present efficient complete algorithms for generating graphs from this class. Finally, we present efficient algorithms for enumerating well-ordered k-degenerate and strongly k-degenerate graphs. This chapter is based on joint work with Reinhard Bauer and Dorothea Wagner [BKW10].

3.1 Preliminary Remarks

The problem of enumerating and generating random graphs has a long tradition in the combinatorial algorithms community. While algorithms for enumerating and generating graphs have applications in exhaustive brute-force algorithms and the experimental analysis of algorithms, the study of these algorithms may also yield new combinatorial insights. The study of analytic combinatorics [FS09], for instance, which is based on an enumerative description of combinatorial objects based on generating functions has provided far-reaching new insights into combinatorial structures and their properties.

When generating random graphs, especially for applications in the experimental analysis of algorithms it is important to provide *complete* generators, that is, generators that generate each graph with positive probability. Otherwise, the experimental analysis will not be representative and, therefore the results of the analysis may be misleading. However, even a complete generator may produce an unwanted bias if the chances of generating a specific subclass of the graphs are very low, while the chances of generating another subclass of the graphs are rather high. Meinert and Wagner [MW11] discuss the impact of various generators for planar graphs on the experimental analysis of planar dominating set algorithms and show that the choice of generator has a significant impact on the outcome of the experimental analysis. Therefore, it is usually desirable to design *uniform* generators, that is, generators that produce each graph from a given class of graphs with equal probability. In cases, where this is not within reach, we can either try to approximate a uniform distribution instead or to design complete generators, whose distribution is as close to a uniform distribution as possible. That is, in general we would like to minimize the maximum difference between the probabilities of generating two different graphs.

Complex objects, such as graphs can be either *unlabeled* or *labeled*, depending on whether we can distinguish between vertices or not. In a labeled graph each vertex has a unique label and vertices are clearly distinguishable, whereas, in an unlabeled graph, vertices are not distinguishable by themselves. Whether we consider the class of labeled or unlabeled graphs has a large impact on whether two graphs are considered to be equal or not. While two unlabeled graphs are considered to be equal if there is an isomorphism between the vertex sets of the graphs, two labeled graphs are considered to be equal only if their edge sets are identical, that is, if there is an edge between two labeled vertices in one of the graphs if and only if there is an edge between two vertices with the same labels in the other graph as well. For instance, two unlabeled paths consisting of three vertices are considered to be equal since there is an isomorphism between the two graphs while there are three different labeled paths as illustrated in Figure 3.1b, one for each label that can be assigned to the center vertex of the path.

We can think of the class of labeled graphs as consisting of a various number of copies of each unlabeled graph. The number of labeled copies of an unlabeled graph with n vertices is exactly the number $n!$ of different labelings divided by the size of the graph's automorphism group, that is, the index of the automorphism group in the symmetric group of the vertices. Consider for instance the class of connected graphs with three vertices and consider a path P consisting of three vertices as well as a triangle T. While, for both graphs, there are exactly $3! = 6$ possible ways of labeling the vertices, all labelings of the triangle will in fact yield the same graph, whereas there are three different labeled copies of the path as illustrated in Figure 3.1. Hence, generating labeled graphs uniformly at random will therefore produce unlabeled objects with small automorphism groups with higher probability, whereas graphs with large automorphism groups, that is, highly symmetric graphs will ge generated less likely.

Ideally, we would like to enumerate and generate unlabeled graphs whenever we are only

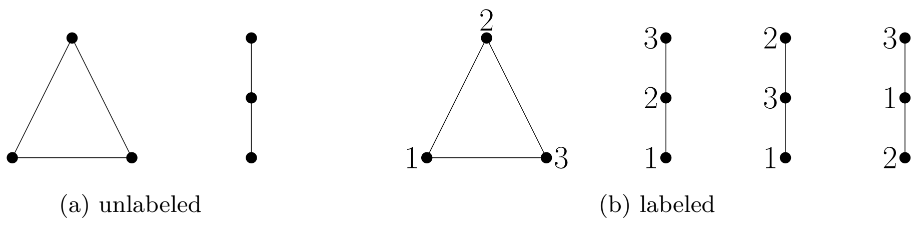


Figure 3.1: Two unlabeled graphs (a) and their labeled counter-parts (b).

interested in the combinatorial structure of the graphs and, thus, rule out all isomorphic copies. However, this is sometimes complicated, especially, if the structure of the objects to be enumerated or generated is rather complex. For instance, to the best of our knowledge, there is as of yet no algorithm for generating unlabeled regular graphs efficiently. In order to overcome the difficulties associated with unlabeled generation, it is common to generate labeled graphs instead, which is often much easier. However, since we are usually interested in unlabeled graphs, even when generating labeled graphs, we only use the labeling for the process of the generation and forget the labels afterwards, thus, associating the labeled copies of the generated graphs with their unlabeled counterparts.

Finally, if the algorithms for enumerating and generating graphs are used for exhaustive algorithms or the experimental analysis of algorithms, they should be fast and easy to implement. Especially, when generating large graphs for experimental analysis we would like to compute large benchmark sets quickly. Thus, even quadratic-time generators may prove to be impracticable. On the other hand, many fast uniform samplers, especially those based on the framework of generating functions, are not easy to implement since the implementation involves complex numerical evaluations and it is often assumed that the computations are performed on a real RAM machine on which the involved numbers can be handled accurately. While the complex combinatorics involved in many generating problems seem to make it unlikely to avoid these computations without sacrificing uniformity of the resulting distribution, the numerical issues involved in these computations must, on all accounts, be resolved and we must find a reasonable balance between the computational efforts and the quality of the resulting distributions in order to obtain practical generators.

3.2 Introduction

In this chapter, we consider the problem of enumerating and generating k-degenerate graphs. A *k-degenerate graph G* is a graph in which every induced subgraph has a vertex with degree at most k. If, in addition, the minimum degree of G is k, then G is a *strongly k-degenerate graph*. The most prominent class of degenerate graphs is the class of planar graphs: Since every induced subgraph of a planar graph is itself planar and since every planar graph contains a vertex of degree at most 5 every planar graph is 5-degenerate. Figure 3.2a shows a 4-degenerate planar graph. Many other well-known classes of graphs are degenerate, including graphs with bounded genus, bounded maximum degree, bounded tree-width as well as minor-free classes of graphs.

The class of k-degenerate graphs plays an interesting role in the theory of fixed parameter

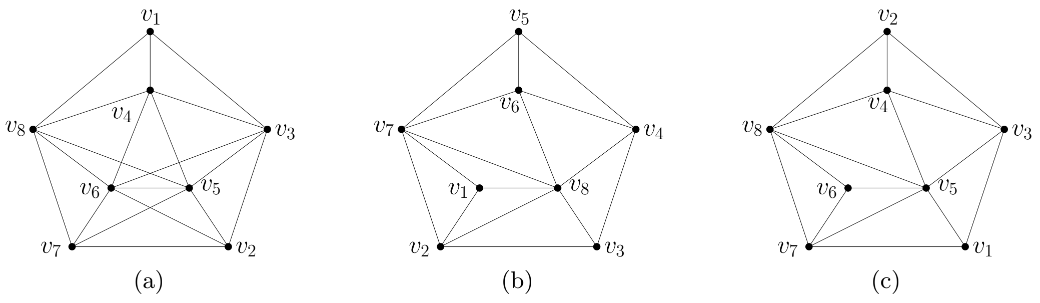


Figure 3.2: Figure (a) shows a well-ordered 4-degenerate graph that is not 3-degenerate and not well-ordered strongly 4-degenerate; Figures (b) and (c) shows two different but isomorphic well-ordered strongly 3-degenerate graph.

tractability since some otherwise $W[2]$-hard domination problems become fixed-parameter tractable for k-degenerate graphs. Noga and Gutner present a linear-time algorithm for computing a dominating set of fixed size in degenerate graphs [AG09] and Golovach and Villanger [GV08] extend this by showing that connected dominating set and dominating threshold set are fixed-parameter tractable for degenerate graphs. Further, Cai et. al use the method of *random separation* to obtain fixed-parameter tractable algorithms for the problem of finding induced cycles and trees in degenerate graphs [CCC06]. However, degenerate graphs have also been studied from a combinatorial point of view by studying packings, colorings and the game chromatic index of degenerate graphs [CZ01, SIN07, BKN08].

It is a well-known fact that the k-degenerate graphs are exactly the graphs whose vertex-set $v_1, \dots, v_n$ can be ordered such that the degree of v_i in the graph induced by the vertices $v_i, \dots, v_n$ is at most k for each i. That is, we can obtain the empty graph by iteratively removing vertices with degree at most k as illustrated in Figure 3.2a. These vertex-sequences are also called *Erdős-Hajnal well-orderings* [SP80] since they were first studied by Erdős and Hajnal [EH66]. *Erdős-Hajnal well-orderings* are closely related to the core decomposition of a graph, a concept that was introduced by Seidman [Sei83] and that has since been used in the analysis of social networks [BZ11]. The *coreness* of a vertex v in the graph G is the minimum integer c such that we can remove v from G by iteratively deleting vertices with degree at most c and the coreness of the graph is the maximum coreness over all vertices in the graph. Thus, the graphs with coreness k are k-degenerate. For example, the coreness of the graph in Figure 3.2a is four while the coreness of the graph in Figure 3.2b is three. The k-shell of a graph consists of all vertices whose coreness is exactly k. Note that these vertices must necessarily have a minimum degree of k after iteratively removing all vertices with degree at most $k - 1$. Thus, the k-shell of a graph is a strongly k-degenerate subgraph. For example, the 3-shell of the graph in Figure 3.2a consists of the single vertex v_1, whereas its 4-shell consists of the remaining vertices. The strongly k-degenerate graphs are the building blocks of the core hierarchy of a graph and thus have applications in graph generators with pre-defined core-hierarchy [BGG$^+$08].

A *labeled* k-degenerate graph with vertex labels $1, \dots, n$ is called *well-ordered* if each vertex with label i is incident to at most k vertices with label greater than i. Figures 3.2b and 3.2c show two different but isomorphic well-ordered k-degenerate graphs. Given a thus labeled vertex-set and its induced order, we consider the problem of enumerating and generating such well-ordered k-degenerate graphs uniformly at random. By generating well-ordered k-degenerate graphs we generate at least one labeled copy of each *unlabeled* k-degenerate graph and we filter some but not all isomorphies compared to the classical labeled approach since not all labelings correspond to Erdős-Hajnal sequences. Thus, by generating a subset of the labeled graphs and filtering out some of the isomorphies, we do in fact decrease the probability of obtaining some of the graphs in favor of others. As an example, consider a star-shaped 1-degenerate graph S_n consisting of a single central vertex v that is connected to $n-1$ other vertices. There are n different labeled copies of this graph, since each label can be assigned to the central vertex v to yield a labeling that is different from all labelings assigning a different number to v. However, there are only two valid Erdős-Hajnal sequences, namely the ones in which the central vertex is the last or the second-last vertex. Although the star-shaped graph is highly symmetric and, thus has a rather large automorphism group, this example already shows that this approach is capable of filtering out a large number of isomorphies.

On the other hand, consider the graph I_n consisting of n isolated vertices. Clearly, this graph has only one labeled copy. Therefore, the class of labeled 1-degenerate graphs contains one labeled copy of I_n but n labeled copies of S_n. Therefore any uniform generator for labeled 1-degenerate graphs will generate I_n proportional to 1, whereas it generates S_n proportional to n. In contrast to this, the class of well-ordered 1-degenerate graphs contains one well-ordered copy of I_n and only two labeled copies of S_n. Thus, a uniform generator for well-ordered 1-degenerate graphs generates S_n proportional to 2. Note that the class $\mathcal{C}$ of k-degenerate graphs with n vertices contains both S_n and I_n for all $k \geq 1$. Therefore, any uniform generator for labeled k-degenerate graphs with n vertices will produce an unlabeled copy of S_n with a probability that is n times higher than the probability of generating I_n, whereas a uniform generator for well-ordered k-degenerate graphs will produce S_n with a probability that is only twice as high as the probability of generating I_n. In general, if there are m isomorphic labeled copies of a graph G in $\mathcal{C}$, then a uniform generator for labeled k-degenerate graphs will generate an unlabeled copy of G with probability m times as high as the probability of generating I_n. Therefore, filtering isomorphies by considering well-ordered k-degenerate graphs instead of ordinary labeled graphs will decrease the difference between the distribution on the unlabeled graphs and their labeled counterparts for k-degenerate graphs. Therefore this method of generation is preferable to the classic labeled generation in terms of the resulting distribution of the networks for k-degenerate graphs.

Previous Work While, to the best of our knowledge, the enumeration an generation of degenerate graphs has not been considered before to its full extent, there has been some effort to obtain generators and enumeration algorithms for various subclasses of degenerate graphs. Planar graphs, for instance, a subset of the 5-degenerate graphs, have been widely studied with respect to random generation. Denise et al. [DVW96] present a Markov process whose stationary distribution is the uniform distribution over all planar subgraphs of a graph. Hence, this process can be used to approximate a uniform distribution of the planar graphs by running the process on a complete graph. Bodirsky et al. [BGK07] present an algorithm for generating labeled planar graphs based a recursive decomposition and Fusy [Fus09] presents a linear-time sampler for generating labeled planar graphs based on generating functions. Additionally, Brinkmann and McKay [BM07] present a fast algorithm for generating unlabeled planar graphs.

Further research has been concentrated on the study of random generation of regular graphs, which is also a subset of the degenerate graphs. According to Gropp [Gro92] the enumeration of regular graphs is a problem that can be traced back to the Dutch mathematician de Vries in 1891. A survey of the results obtained for regular graphs is given by Wormald [Wor99]. Kim and Vu [KV03] analyze a generator for random regular graphs given previously by Steger and Wormald [SW99] and prove that it produces random regular graphs asymptotically uniformly. Read and Wormald [RW06] study the number of labeled 4-regular graphs, a problem that is closely related to enumeration and uniform generation, and Ding et al. [DKS09] study the problem of generating 5-regular planar graphs.

Contribution We investigate the problem of generating well-ordered k-degenerate graphs uniformly at random and present fast algorithms whose running time is almost optimal after a precomputation in the real-RAM machine model. Since the precomputation must only be performed once for the uniform samplers it can be amortized when generating a large set of graphs. Further, we present practical and easy-to-implement complete generators for k-degenerate graphs whose running time is linear in the size of the generated graph in the RAM machine model and, thus, optimal. For well-ordered k-degenerate graphs with given number of vertices, this algorithm is based on an asymptotic approximation of the uniform distribution of these graphs.

Additionally, we consider strongly k-degenerate graphs, which are a natural generalization of k-regular graphs with applications in the design of generators for graphs with given core hierarchy [BGG$^+$07]. We present fast and easy-to-implement complete algorithms for this class of graphs. Finally, we consider the problem of enumerating k-degenerate and strongly k-degenerate graphs and we provide efficient algorithms whose amortized running time is polynomial in the size of the generated graphs. Since strongly k-degenerate graphs are a generalization of k-regular graphs, the presented algorithms can be used to generate k-regular graphs without further modification.

Content In Section 3.3 we review some basic concepts related to random variate generation, such as the inversion method, and we present a random variate generator for the k-restricted binomial distribution, a distribution that appears during the uniform generation of well-ordered k-restricted graphs with a given number of vertices. In Section 3.4 we present an approximate random variate generator for the k-restricted binomial distribution that can be used to generate well-ordered k-degenerate graphs with approximately uniform distribution. These first two sections establish preliminaries and auxiliary results in probability theory that will be used in the following sections.

In Section 3.5 we present uniform as well as faster complete non-uniform generators for well-ordered k-degenerate graphs. Subsequently, Section 3.6 is concerned with algorithms for generating well-ordered strongly k-degenerate graphs. In this section we present fast and complete non-uniform generators for the class of well-ordered strongly k-degenerate graphs. Finally, we present efficient algorithms for enumerating all well-ordered k-degenerate and strongly k-degenerate graphs in Section 3.7. A summary of the contributions can be found in Table 3.1.

Table 3.1: Summary of the contributions in this chapter. If not otherwise marked the running times are in the classical RAM machine model.

		well-ordered k-degenerate		well-ordered strongly k-degenerate	
sampling					
uniform	n	$\mathcal{O}(n \log k + m)^*$	[Th. 3.4]	open	
	(n, m)	$\mathcal{O}(n \log k + m + nk)^*$	[Th. 3.6]	open	
non-uniform	n	$\mathcal{O}(n + m)$	[Th. 3.5]	$\mathcal{O}(n + m)$	[Th. 3.9]
	(n, m)	$\mathcal{O}(n + m)$	[Th. 3.7]	$\mathcal{O}(nmk + n \log k)$	[Th. 3.8]
enumeration					
enumeration	n	$\mathcal{O}(nm + m^2)$	[Th. 3.11]	$\mathcal{O}(nm + m^2)$	[Th. 3.11]
	(n, m)	$\mathcal{O}(nm + m^2)$	[Th. 3.11]	$\mathcal{O}(n^{3/2} m^2)$	[Th. 3.12]

*real-RAM machine model

3.3 Random Variate Generator for the Restricted Binomial Distribution

In this section we introduce the k-restricted binomial distribution that we will use for the generation of well-ordered k-degenerate graphs in Section 3.5 and we study random variate generators for this distribution. First we show how to generate random variates for the k-restricted binomial distribution using the inversion method [Dev86]. By pre-computing and re-using data needed to apply the inversion method, we can improve over the classical inversion method when generating random variates for k-restricted distributions with different parameters. Finally, we show how to approximate the k-restricted binomial distribution using the standard normal distribution and we show how this approximation can be turned into a random variate generator whose distribution is approximately k-restricted binomial and whose running time is constant in the classical RAM machine model. Before we study the k-restricted binomial distribution, however, we provide some basic concepts related to distributions and random variate generation that will only be used in this chapter.

3.3.1 Preliminaries

The following definitions are based on the textbook by Klenke [Kle08] and the book by Devroye [Dev86]. Let Ω be a non-empty set and let $\mathcal{A} \subseteq 2^{\Omega}$. Then $\mathcal{A}$ is a *σ-algebra* if $\Omega \in \mathcal{A}$ and $\mathcal{A}$ is closed both under complements and countable unions. If $\mathcal{A}$ is a σ-algebra, then the tuple $(\Omega, \mathcal{A})$ is called a *measurable space* and the sets in $\mathcal{A}$ are called *measurable.* If Ω is at most countably infinite and if $\mathcal{A} = 2^{\Omega}$, then $(\Omega, 2^{\Omega})$ is called *discrete.* A function $\mu\colon \mathcal{A} \to [0, \infty]$, where $\mathcal{A}$ is a σ-algebra, is called σ-additive, if

$$\mu\left(\biguplus_{i \in I} A_i\right) = \sum_{i \in I} \mu(A_i)$$

for any choice of countably many pairwise disjoint set $A_i \in \mathcal{A}$ for $i \in I$. If $\mathcal{A}$ is a σ-algebra and $\mu\colon \mathcal{A} \to [0, \infty]$ is a σ-additive function with $\mu(\emptyset) = 0$, then μ is called a *measure*. If, in addition, $\mu(\Omega) = 1$, then μ is called a *probability measure* and the triple $(\Omega, \mathcal{A}, \mu)$ is called a *probability space*. Further, if $(\Omega, \mathcal{A})$ is discrete, then the triple $(\Omega, \mathcal{A}, \mu)$ is called a *discrete probability space*. If, additionally, Ω is finite, we call the triple a *finite discrete probability space*. For a probability space $(\Omega, \mathcal{A}, \mu)$, the sets $A \in \mathcal{A}$ are called *events*.

If $(\Omega, \mathcal{A})$ and $(\Omega', \mathcal{A}')$ are two measurable spaces and $X\colon \Omega \to \Omega'$ is a map such that $X^{-1}(A') \in \mathcal{A}$ for any $A' \in \mathcal{A}'$, then X is called a *measurable map*. Let $(\Omega, \mathcal{A}, \mathbb{P})$ be a probability space. Intuitively, a *random variable* is a variable that is associated with the outcome of a random experiment and that describes an observation A' associated with an event in A. Formally, a random variable is a measurable map $X\colon \Omega \to \Omega'$ that maps the events in $\mathcal{A}$ to a space of possible observations $\mathcal{A}'$. If $(\Omega, \mathcal{A}) = (\mathbb{R}, \mathcal{B}(\mathbb{R}))$, where $\mathcal{B}(\mathbb{R})$ denotes the Borel σ-algebra on $\mathbb{R}$ that is generated by all intervals on $\mathbb{R}$, then X is called a *real random variable*. If not otherwise stated, we will assume that random variables are real. For $A' \in \mathcal{A}'$ we write $\{X \in A'\} := X^{-1}(A')$ and $\mathbb{P}(X \in A') := \mathbb{P}(X^{-1}(A'))$. Further, we write $\{X \geq a\} := X^{-1}([a, \infty))$ and $\{X \leq b\} := X^{-1}((-\infty, b])$ for $a, b \in \mathbb{R}$.

Let X be a random variable. Then $\mathbb{P}_X := \mathbb{P} \circ X^{-1}$ is called the *distribution of* X. The distribution of a discrete probability space is called *discrete*, otherwise it is called *continuous*. If X is a real random variable, then the map $F_X\colon x \mapsto \mathbb{P}(X \leq x)$ for $x \in \mathbb{R}$ is called the *(cumulative) distribution function of* X and we write $X \sim \mathbb{P}_X$ and say that X has distribution $\mathbb{P}_X$. Further, if it exists, the derivative of the distribution function $f_X(x) = dF_X(x)/dx$ is called the *(probability) density function of* X. Note that the distribution function of a discrete function is defined on $\mathbb{R}$. That is, the distribution function of a discrete real random variable X is defined for values outside the domain of X. If X is a real random variable and if f_X exists, then the *expected value* is defined as

$$\mathbb{E}(X) = \int_{-\infty}^{\infty} x f_X(x) dx \; .$$

If X is a discrete random variable, then the expected value is defined as

$$\mathbb{E}(x) = \sum_{i=1}^{\infty} x \mathbb{P}(X = x_i) \; ,$$

where x_i $(i = 1, 2, \dots)$ denote the at most countably many values that X can assume. The *variance* of X is defined as $\operatorname{Var}(X) = \mathbb{E}(X^2) - \mathbb{E}(X)^2$. Throughout the chapter we use the discrete and continuous distributions summarized in Table 3.2. We define the *k-restricted binomial distribution* as the discrete probability distribution with domain $\{1, \dots, k\}$ and distribution function

$$F_{n,k}(x) = \frac{\sum_{i=0}^{x} \binom{n}{i}}{\sum_{i=0}^{k} \binom{n}{i}} \; .$$

While the standard binomial distribution $\operatorname{Binom}(n, \frac{1}{2})$ is the distribution of the number of successes in a sequence of n independent random experiments whose outcome is either success or failure, each occurring with probability $\frac{1}{2}$, the k-restricted binomial distribution models a similar distribution restricted to at most k successes in total. Hence, the n-restricted binomial

Table 3.2: Distributions used in this chapter.

Distribution	Notation	Domain	Density
discrete			
Uniform	$\text{Uniform}(X)$	$x \in X,\ \lvert X\rvert < \infty$	$f(x) = 1/\lvert X\rvert$
Binomial	$\text{Binom}(n, p)$	$x \in \{0, \dots, n\}$	$f(x) = \binom{n}{i} p^i (1-p)^{n-i}$
k-restricted Binomial	$f(x) = \text{Binom}_{\leq k}(n)$	$x \in \{0, \dots, k\}$	$f(x) = \binom{n}{x} \cdot \left(\sum_{i=0}^{k} \binom{n}{k}\right)^{-1}$
continuous			
Uniform	$\text{Uniform}([a, b])$	$x \in [a, b]$	$f(x) = 1/(b-a)$
Normal	$\text{Normal}(\mu, \sigma^2)$	$x \in \mathbb{R}$	$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \cdot \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$

distribution is identical to the standard binomial distribution with parameters n and $p = \frac{1}{2}$. The remaining distribution in Table 3.2 are well-known standard distributions.

A *random variate* is a real number that is the output of a randomized algorithm that halts with probability one. Such an algorithm is also called a *random variate generator*. If $\mathcal{G}$ is a random variate generator that produces random values in $[0, 1]$ with uniform distribution, then the resulting random variate is called a *random uniform* $[0, 1]$*-variate*. We assume that we can store and manipulate real numbers and that there exists a random uniform $[0, 1]$-variate generator that we may use in order to obtain random numbers.

A graph $G = (V, E)$ is called *labeled* if each vertex $v \in V$ is assigned a unique label $\lambda(v) \in \mathbb{N}$, that is, $\lambda \colon V \to \mathbb{N}$ is an injective function. Throughout the chapter we assume that $V = \{v_1, \dots, v_n\}$ and that $\lambda(v_i) = i$ for all $1 \leq i \leq |V|$. Two labeled graphs $G = (V, E)$ and $G' = (V', E')$ are *identical* if and only if $|V| = |V'|$ and $\{\lambda^{-1}(i), \lambda^{-1}(j)\} \in E$ if and only if $\{\lambda'^{-1}(i), \lambda'^{-1}(j)\} \in E$ for all $1 \leq i, j \leq |V|$, where λ and λ' denote the labels of G and G', respectively.

Throughout this chapter, we only consider undirected loopless graphs without multiple edges between any pair of vertices. The *neighborhood* $N(v_i)$ of a vertex v_i is the set of vertices v_j such that there is an edge $\{v_i, v_j\} \in E$. The degree of a vertex v_i in G is denoted by $\mathrm{d}_G(v_i)$. If the context is clear we will omit the subscript G. For a vertex v_i we define the *successor-degree* by

$$\mathrm{d}^+(v_i) := |\{\{v_i, v_j\} \in E \mid i < j\}|$$

as well a the *predecessor-degree* by

$$\mathrm{d}^-(v_i) := |\{\{v_i, v_j\} \in E \mid j < i\}| \,.$$

A *well-ordered k-degenerate graph* $G = (V, E)$ is a labeled k-degenerate graph with n vertices $v_1, \dots, v_n$ such that $\lambda(v_i) = i$ and such that $\mathrm{d}^+(v_i)$ is bounded by $\min\{n - i, k\}$ for all $1 \leq i \leq n$. Let G be a well-ordered k-degenerate and let G' be the subgraph induced by the vertices $v_i, \dots, v_n$ for $i > 1$. By definition, G' is not well-ordered k-degenerate, but by re-labeling the vertices using the mapping $j \mapsto j - i + 1$ we obtain a well-ordered k-degenerate graph. We will not always mention the re-labeling explicitly. Instead we will say that G' is a well-ordered k-degenerate graph with vertices $v_i, \dots, v_n$. We denote the class of well-ordered k-degenerate graphs with n vertices by $\mathcal{D}(n,k)$ and its cardinality by $D(n,k)$. Accordingly, we denote the class of well-ordered k-degenerate graphs with n vertices and m edges, by $\mathcal{D}(n,m,k)$ and its cardinality by $D(n,m,k)$.

With some abuse of notation we denote the set of all s-element subsets of a set X by $\binom{X}{s}$. Let X be a finite set and let $f : X \to \mathbb{R}$ be a function assigning a real value to each of given set of objects in X. Choosing an element $x \in X$ *proportional to* f means choosing x with probability

$$p_x := \frac{f(x)}{\sum_{z\in X} f(z)} \,.$$

The term $G[u_1, \dots u_k]$ is used as a shorthand for $G[\{u_1, \dots, u_k\}]$ and refers to the subgraph of G induced by the vertices $u_1, \dots u_k$.

3.3.2 Inversion Method

The inversion method [Dev86] is a well-known method for generating random variates with arbitrary continuous distribution. This method is based on the observation that the inverse of the distribution function of a given continuous distribution Θ can be used to transform a random uniform $[0,1]$-variate into a random variate with distribution Θ. To generate a random variate with distribution Θ with distribution function F, we simply generate a random uniform $[0,1]$-variate U and compute $F^{-1}(U)$ as summarized in Algorithm 3.1. The correctness of this algorithm is established by the following theorem.

Theorem 3.1 ([Dev86])**.** *Let F be a continuous distribution function on $\mathbb{R}$ with inverse F^{-1} defined by*

$$F^{-1}(u) = \inf\{x \in \mathbb{R} \mid F(x) = u\}$$

for $0 < u < 1$. If U is a uniform $[0,1]$ random variable, then $F^{-1}(U)$ has distribution function F. Also, if X has distribution function F, then $F(X)$ is uniformly distributed on $[0,1]$.

A similar concept can be used for the discrete case and is also shortly mentioned in [Dev86]. Let $(\Omega, 2^{\Omega}, \mathbb{P})$ be a finite discrete probability space and let X be a real random variable. Then X can assume a finite set of values $x_1 < x_2 < \cdots < x_r \in \mathbb{R}$ such that $\mathbb{P}(x_i) > 0$ for all $1 \le i \le r$. Setting $\mathbb{P}(x_i) =: p_i$, the cumulative distribution function of X is given by

$$F(x) = \mathbb{P}(X \le x) = \sum_{x_i \le x} p_i \tag{3.1}$$

Algorithm 3.1: Continuous Inversion Method

```
Input: Continuous distribution function $F\colon \mathbb{R} \to [0,1]$
Output: Random variate $X$ with distribution $F$
1 Generate a uniform $[0,1]$ random variate $U$
2 $X \leftarrow F^{-1}(U)$
3 return $X$
```

Algorithm 3.2: Discrete Inversion Method

```
Input: $x_1, \dots, x_r \in \mathbb{R}$, distribution function $F\colon \{x_1, \dots, x_r\} \to [0,1]$
Output: Random variate $X$ with distribution $F$
1 Compute and store $F(x_i)$ for $1 \leq i \leq r$
2 Generate a uniform $[0,1]$ random variate $U$
3 compute $X := F^{-1}(U)$ by binary search on $F(x_i)$ $i = 1, \dots n$ using Equation (3.4)
4 return $X$
```

Let the inverse of F be defined as the function $F^{-1}\colon [0,1] \to \{x_1, \dots, x_r\}$ such that

$$F^{-1}(p) = \min\{x \in \{x_1, \dots, x_r\} \mid F(x_i) \geq p\} \tag{3.2}$$

for $p \in [0,1]$. Further, let $P_i := \sum_{x_j \leq x_i} p_i$ be the values that F can assume. Then clearly, $F^{-1}(P_i) = x_i$ and, therefore, we have

$$F(F^{-1}(P_i)) = P_i \text{ and } F^{-1}(F(x_i)) = x_i \tag{3.3}$$

for all $1 \leq i \leq r$. Further, note that, if $x_i = F^{-1}(p)$ for $i > 1$, then we have

$$F(x_{i-1}) < p \leq F(x_i)\ . \tag{3.4}$$

Additionally, we have $0 < p \leq F(x_1)$ for $x_1 = F^{-1}(p)$. That is, we can compute the inverse F^{-1} of a discrete distribution function using binary search given the sorted sequence of the values $F(x_i)$ for $1 \leq i \leq r$. Let U be a random uniform $[0,1]$-variate and let $Y := F^{-1}(U)$. Then

$$\mathbb{P}(Y = x_i) = \mathbb{P}(F(x_{i-1}) < U \leq F(x_i)) = F(x_i) - F(x_{i-1}) = p_i\ . \tag{3.5}$$

Hence, Y is a random variate with distribution function F. In order to generate a random variate of a discrete distribution with given distribution function F we can therefore proceed as summarized in Algorithm 3.2. First, we compute the values $F(x_1), \dots, F(x_r)$ corresponding to the cumulative probabilities of F for the values $x_1, \dots, x_r$ that the distribution may assume. We store these values in an array sorted according to the order of $x_1, \dots, x_r$. Then we generate a random uniform $[0,1]$-variate U and compute $F^{-1}(U)$ according to Equation (3.4) by binary search on the stored values. Assuming that we can evaluate F in constant time, we can pre-compute and store the values $F(x_1), \dots, F(x_r)$ in time and space $\mathcal{O}(r)$ and we can compute $F^{-1}(U)$ in time $\mathcal{O}(\log r)$. The correctness of this approach is based on Equation (3.5). This result is summarized by the following lemma.

Lemma 3.1. *Let Y be a discrete random variable with distribution function F and let D be the domain of Y. Further, assume that $F(x)$ can be computed in $\mathcal{O}(1)$ time for all $x \in D$. Then we can generate a random variate X with domain D and distribution function F in time $\mathcal{O}(\log |D|)$ after precomputing and storing $|D|$ values using $\mathcal{O}(|D|)$ time and space, respectively.*

3.3.3 The Restricted Binomial Distribution

If the discrete inversion method is used more than once to generate random variates with the same distribution, then it suffices to perform Line 1 of Algorithm 3.2 only once, as a pre-computation. However, if the discrete inversion method is used to generate random variates with different distributions, then the precomputation must, in general, be performed for each of the distributions. As an application of the discrete inversion lemma we generate random k-restricted binomial variates for different values of k. Although the distribution functions differ for different values of k, they are identical to a prefix of the standard binomial distribution up to normalization. In fact, we can use this observation to obtain a random variate generator for random variates $X \sim \mathrm{Binom}_{\le k}(n)$ for all $1 \le k \le n$ that pre-computes and stores only n values. Clearly, this outperforms the $\sum_{i=1}^{n} i = \Omega(n^2)$ precomputation time and space needed for independent random variate generators.

Lemma 3.2. *Let n be a fixed integer and let Y be a k-restricted random variable for $0 \le k \le n$ with distribution function $F_{n,k}$. Then we can generate a random variate X with domain $D = \{0, \dots, k\}$ with distribution function $F_{n,k}$ in $\mathcal{O}(\log k)$ time using $\mathcal{O}(n)$ precomputation time and space.*

Proof. In a precomputation we compute and store the prefix-sums of the binomial coefficients $B_n(i) := \sum_{j=0}^{i} \binom{n}{j}$ for all $0 \le i \le n$. This can be done in linear time in our model of computation since

$$\binom{n}{i+1} = \frac{n-i}{i+1}\binom{n}{i} \, . \tag{3.6}$$

Hence, we can iteratively obtain B_{i+1} and $\binom{n}{i+1}$ from B_i and $\binom{n}{i}$ in constant time. This implies that $F(i) = \mathbb{P}(Y \le i)$ can be computed from $F(i) = B_n(i)/B_n(k)$ in constant time. Hence, the binary search from Lemma 3.1 can be performed in time $\mathcal{O}(\log k)$ on the values $B_n(i)$. $\square$

As mentioned, the linear-time precomputation needs only be performed once and the linear overhead is amortized as soon as we sample $\Omega(n)$ values from this distribution. Although there are other methods for sampling discrete random variates with arbitrary distributions, such as the table-lookup methods or methods based on hashing [Dev86], we prefer the described inversion methods for two reasons. On the one hand, it provides a reasonably good worst case guarantee on the running time and on the other hand, the space requirement is reasonably small. For the applications in this chapter, methods based on table-lookup are infeasible due to the large numbers involved and methods based on hashing do not give any guarantees. We note, however, that methods based on hashing may indeed be used to achieve expected constant time random variate generation after a suitable pre-computation.

3.4 Approximating the Restricted Binomial Distribution

In the previous section, we have seen how to generate k-restricted binomial random variates using a discrete inversion method. A major drawback of this approach is the additional linear space for the pre-computation and the logarithmic factor on the running time due to the binary search. Additionally, the large numbers involved in the computations make the algorithms rather infeasible for large n. While this is not an issue within the real-RAM machine model in theory, we have to cope with this issue in practical applications. Since exact arithmetic involving large numbers is computationally costly, we therefore propose approximating the distributions instead. In this section we will study an approximation of the k-restricted binomial distribution and a corresponding random variate generator with applications in the generation of well-ordered k-degenerate graphs. Our goal is to obtain an approximation with a provable guarantee on the maximal deviation from the uniform distribution that can be used as the basis for further numerical approximation. We note that numerical approximations based, for instance, on Taylor series expansion and Padé approximants, cannot be computed for the involved distributions directly, since the involved functions do not allow to be described by a closed formula.

Overview In order to generate random variates with approximately k-restricted binomial distribution we proceed in several steps. First, we show that we can generate k-restricted binomial random variates using a modification of the discrete inversion method on the standard binomial distribution $\mathrm{Binom}(n, \frac{1}{2})$ in Lemma 3.3. In order to make the inversion method efficient, we must find an efficient way of evaluating the binomial distribution function and its inverse. Since easy-to-evaluate closed formulae for these functions are unlikely to exist, we resort to approximating these functions instead. In Lemma 3.4 we show that we can approximate the binomial distribution function using the standard normal distribution function. This is established by the Berry-Esseen Theorem [Ber41, Ess42, Ess56, KS10]. Next, we show that the inverse of the normal distribution can, in turn, be used in order to approximate the inverse of the binomial distribution function in Lemma 3.5. The error of this approximation deteriorates towards zero and one, but it is quite useful in an interval that is not too close to these values. Since we cannot evaluate the normal distribution function efficiently, we conclude by establishing the error of additionally approximating the normal distribution function numerically in Lemma 3.6. Then we present an easy-to-implement random variate generator based on the numerical approximations. We start with the following lemma, which shows how to generate a random variate with k-restricted binomial distribution using a modification of the inversion method and the standard binomial distribution.

Lemma 3.3. *Let $n \in \mathbb{N}$ and let F_n be the distribution function of a random variable with distribution $\mathrm{Binom}(n, \frac{1}{2})$. Further, let U be a random uniform $[0, p]$-variate for $p = F_n(k)$ with $0 \le k \le n$. Then $X = F_n^{-1}(U)$ is a random variate with distribution $\mathrm{Binom}_{\le k}(n)$, where F_n^{-1} is defined according to Equation 3.2.*

Proof. Recall that $X = F_n^{-1}(U)$ satisfies

$$F_n(X-1) < U \le F_n(X) \tag{$\star$}$$

by Equation (3.4) since X assumes only integral values. Then we have

$$\mathbb{P}(X = i) = \mathbb{P}\left(F_n(i-1) < U \le F_n(i)\right) = \frac{F_n(i) - F_n(i-1)}{p},$$

due to ($\star$) and since U is a random uniform $[0, p]$-variate, which simplifies to

$$\mathbb{P}(X = i) = \frac{2^{-n}\binom{n}{i}}{F_n(k)}$$

since $p = F_n(k)$ and since

$$F_n(i) - F_n(i-1) = \sum_{j=0}^{i} 2^{-n}\binom{n}{j} - \sum_{j=0}^{i-1} 2^{-n}\binom{n}{j} = 2^{-n}\binom{n}{i} .$$

This yields

$$\mathbb{P}(X = i) = \frac{\binom{n}{i}}{\sum_{j=0}^{k}\binom{n}{j}} .$$

Hence, X is a k-restricted binomial random variate with distribution $\mathrm{Binom}_{\leq k}(n)$ by the definition of this distribution. □

This suggests, that we can use the inversion method on the standard binomial distribution using a truncated uniform random variate as a starting point for generating k-restricted binomial random variates. In order to make this approach efficient, however, we must find an adequate way of approximating the binomial distribution function and its inverse, since it seems unlikely that there is an efficient way of computing the exact values of this distribution efficiently. For our approximation we first show that the binomial distribution and its inverse can be approximated using the standard normal distribution and its inverse, respectively. This is possible by considering $X \sim \mathrm{Binom}(n, \frac{1}{2})$ as the sum of n independent random Bernoulli variables. The key to assessing the quality of this approximation is given by the Berry-Esseen-Theorem [Ber41, Ess42, Ess56, KS10].

Theorem 3.2 (Berry-Esseen [Ber41, Ess42, Ess56, KS10]). *Let $(X_n)_{n\geq 1}$ be a sequence of independent identically distributed random variables with $a = \mathbb{E}(X_1)$, $0 < \sigma^2 = V(X_1) < \infty$ and $\mathbb{E} \mid X_1 \mid^3 < \infty$. Let F_n denote the distribution function of the random variable $Z := (\sigma\sqrt{n})^{-1} \cdot (\sum_{j=1}^{n} X_j - na)$ and let Φ denote the distribution function of the normal distribution. Then*

$$\sup_{x\in\mathbb{R}} |F_n(x) - \Phi(x)| \leq \frac{C}{\sqrt{n}} \cdot \mathbb{E}\left|\frac{X_1 - a}{\sigma}\right|^3$$

where C is an absolute constant fulfilling $0.4097 \approx (\sqrt{10} + 3)/(6\sqrt{2\pi}) \leq C < 0.4784$.

In other words, the error of approximating the sum of n independent identically distributed random variables by the standard normal distribution can be bounded by an error term that is approaching zero as n tends to infinity. In the following we let C denote the constant of the Berry-Esseen Theorem. The Berry-Esseen Theorem immediately implies the following lemma, which provides the bound for the case that the considered random variables have a Bernoulli distribution, that is, the sum of these variables has a binomial distribution.

Lemma 3.4. *Let $n \in \mathbb{N}$, let X be a random variable with $X \sim \mathrm{Binom}(n, \frac{1}{2})$, let $F_n(x)$ be the distribution function of X and let Φ denote the distribution function of the standard normal distribution. Then*

$$\sup_{x\in\mathbb{R}} \left|F_n(x) - \Phi\left(\frac{2x-n}{\sqrt{n}}\right)\right| \leq \frac{C}{\sqrt{n}} .$$

Proof. We have $X \sim \text{Binom}(n, \frac{1}{2}) \sim X_1 + \ldots + X_n$ with $X_i \sim \text{Binom}(1, \frac{1}{2})$. Since $X_1 \sim \text{Binom}(1, \frac{1}{2})$ we have $a = \mathbb{E}|X_1| = \frac{1}{2}$ and $\sigma^2 = V(X_1) = \frac{1}{4}$. Let

$$\begin{aligned} Z &:= (\sigma\sqrt{n})^{-1} \cdot \left(\sum_{j=1}^{n} X_j - na \right) \\ &= 2\sqrt{n}^{-1} \cdot \left(\sum_{j=1}^{n} X_j - \frac{n}{2} \right) && \text{since } a = \frac{1}{2} \text{ and } \sigma = \frac{1}{2} \\ &= 2\sqrt{n}^{-1} \cdot \left(X - \frac{n}{2} \right) . \end{aligned}$$

Hence, Z is the random variable obtained by centering and scaling X. Let G_n be the distribution function of Z. Then the previous equation implies

$$G_n(z) = \mathbb{P}(Z \leq z) = \mathbb{P}\left(X \leq \frac{1}{2}\left(\sqrt{n} \cdot z + n\right) \right).$$

Using Theorem 3.2 we obtain

$$\sup_{z\in\mathbb{R}} |G_n(z) - \Phi(z)| = \sup_{z\in\mathbb{R}} \left| \mathbb{P}\left(X \leq \frac{1}{2}\left(\sqrt{n} \cdot z + n\right) \right) - \Phi(z) \right| \leq \frac{C}{\sqrt{n}} \cdot \mathbb{E}\left| \frac{X_1 - a}{\sigma} \right|^3 .$$

We have $\mathbb{E}\left|\frac{X_1-a}{\sigma}\right|^3 = \mathbb{E}\left|2X_1 - 1\right|^3 \leq 1$ since $-1 \leq 2X_1 - 1 \leq 1$. Then the previous equation yields

$$\begin{aligned} \sup_{x\in\mathbb{R}} \left| F_n(x) - \Phi\left(\frac{2x-n}{\sqrt{n}} \right) \right| &= \sup_{x\in\mathbb{R}} \left| \mathbb{P}(X \leq x) - \Phi\left(\frac{2x-n}{\sqrt{n}} \right) \right| \\ &= \sup_{z\in\mathbb{R}} \left| \mathbb{P}\left(X \leq \frac{1}{2}\left(\sqrt{n} \cdot z + n\right) \right) - \Phi(z) \right| \\ &= \sup_{z\in\mathbb{R}} |G_n(z) - \Phi(z)| \\ &\leq \frac{C}{\sqrt{n}} . \end{aligned}$$

□

The previous lemma provides an approximation guarantee for approximating the standard binomial distribution by the standard normal distribution after centering and scaling the arguments. Hence, we can approximate the standard binomial distribution using $F_n(x) \approx \Phi(\zeta(x))$ where

$$\zeta(x) = \frac{2x-n}{\sqrt{n}} .$$

Accordingly, we write

$$\zeta^{-1}(z) = \frac{\sqrt{n}z + n}{2} .$$

The previous lemma shows that $|F_n(x) - \Phi(\zeta(x))|$ converges uniformly to zero as n goes to infinity. In order to apply Lemma 3.3, however, we also need to approximate the inverse $F_n^{-1}(p)$. The following lemma shows that this is possible using the inverse of the standard normal distribution function for large n. Due to the singularities of this function at 0 and 1, however, it is not possible to obtain uniform convergence. Thus, the quality of the approximation deteriorates towards the boundaries of the interval $[0, 1]$. The following lemma provides a family of approximations G_n defined on a symmetric interval $I := [p_0, 1 - p_0]$ around $\frac{1}{2}$ whose approximation guarantee can be bounded in terms of p_0 as n goes to infinity. That is, as n goes to infinity, the error can be bounded by a constant in this interval.

Lemma 3.5. *Let $n \in \mathbb{N}$, let X be a binomially distributed random variable with $X \sim$ $\mathrm{Binom}(n, \frac{1}{2})$ and let $F_n(x)$ be the distribution function of X. Let $P_i := F_n(i)$ denote the values that F_n can assume for $0 \le i \le n$ and let F_n^{-1} be the inverse of F_n as defined in Equation (3.2). Further, let $I := [p_0, 1-p_0]$ for a fixed value $0 < p_0 < \frac{1}{2}$ and let $\ell := \ell(n) := \max\{i \mid P_i < \frac{C}{\sqrt{n}}\}$ and $r := r(n) := \min\{i \mid P_i > 1 + \frac{C}{\sqrt{n}}\}$. Let*

$$G_n(p) = \begin{cases} G_n(P_{\ell+1}) - 1 & p \in (0, P_{\ell+1}) \\ \zeta^{-1}\left(\Phi^{-1}(p)\right) & p \in [P_{\ell+1}, P_{r-1}] \\ G_n(P_{r-1}) + 1 & p \in (P_{r-1}, 1) \end{cases} .$$

Then

$$\lim_{n\to\infty} \sup_{p \in I} \left| F_n^{-1}(p) - G_n(p) \right| \le \sqrt{2\pi} \cdot C \cdot \exp\left\{ \frac{\Phi^{-1}(p_0)^2}{2} \right\} + 1 .$$

Proof. We prove the lemma as follows. First, we define two function G_n^- and G_n^+ and show that these functions are minorants and majorants of G_n and F_n^{-1}, respectively. In order to achieve this, we first show that these functions bound F_n^{-1} from above and below, respectively, on the finite number of values that F_n^{-1} assumes in the interval I. Using this result, we show that the functions bound F_n^{-1} on the whole interval I. Given this, we can bound the error of approximating F_n^{-1} by G_n on I in terms of the difference between G_n^+ and G_n^- since both functions bound both G_n and F_n^{-1} from above and below, respectively.

Assume that n_0 is such that $I \subset (\frac{C}{\sqrt{n_0}}, 1 - \frac{C}{\sqrt{n_0}})$. Then for all $n \ge n_0$ we have $I \subset I_n := (\frac{C}{\sqrt{n}}, 1 - \frac{C}{\sqrt{n}})$. Since we consider the following equations in the limit for $n \to \infty$ we may assume that $n \ge n_0$ henceforth. Note that ℓ and r are chosen in such a way that P_ℓ is the largest probability not in I_n and P_r is the smallest probability not in I_n. Thus, the probabilities $P_{\ell+1}, \ldots, P_{r-1}$ are contained in I_n. We consider the functions

$$G_n^+(p) = \begin{cases} G_n^+(P_{\ell+1}) - 1 & p \in (0, P_{\ell+1}) \\ \zeta^{-1}\left(\Phi^{-1}\left(p + \frac{C}{\sqrt{n}}\right)\right) & p \in [P_{\ell+1}, P_{r-1}] \\ G_n^+(P_{r-1}) + 1 & p \in (P_{r-1}, 1) \end{cases}$$

and

$$G_n^-(p) = \begin{cases} G_n^-(P_{\ell+1}) - 1 & p \in (0, P_{\ell+1}) \\ \zeta^{-1}\left(\Phi^{-1}\left(p - \frac{C}{\sqrt{n}}\right)\right) & p \in [P_{\ell+1}, P_{r-1}] \\ G_n^-(P_{r-1}) + 1 & p \in (P_{r-1}, 1) \end{cases}$$

on domain $(0, 1)$. We will show that these functions are a minorant and a majorant of both G_n and F_n^{-1}, respectively. That is, we can bound the difference between G_n and F_n^{-1} by the difference of G_n^- and G_n^+. Since G_n is defined in terms of the inverse of the standard normal distribution function this provides a bound on the quality of the respective approximation.

Note that both functions are well-defined on $(0, 1)$ since $[P_{\ell+1}, P_{r-1}] \subseteq I_n$ and since

$$0 < p - \frac{C}{\sqrt{n}} \leq p + \frac{C}{\sqrt{n}} < 1$$

for all $p \in I_n$. Further, note that both G_n^- and G_n^+ are non-decreasing since $\zeta^{-1} \circ \Phi^{-1}$ is non-decreasing. Clearly, G_n^- is a minorant of G_n and G_n^+ is a majorant of G_n on I_n. Next, we show that G_n^- is a minorant and G_n^+ is a majorant of F_n^{-1} on I, respectively. According to the bound

$$\sup_{x\in\mathbb{R}} |F_n(x) - \Phi(\zeta(x))| \leq \frac{C}{\sqrt{n}}$$

provided in Lemma 3.4 we have

$$F_n(i) - \frac{C}{\sqrt{n}} \leq \Phi(\zeta(i)) \leq F_n(i) + \frac{C}{\sqrt{n}} \tag{3.7}$$

due to Lemma 3.4 for all $i \in \{0, \ldots, n\}$ such that $P_i \in I_n$. Since $\zeta^{-1} \circ \Phi^{-1}$ is non-decreasing, this is equivalent to

$$G_n^-(P_i) = \zeta^{-1}\left(\Phi^{-1}\left(P_i - \frac{C}{\sqrt{n}}\right)\right) \leq i \leq \zeta^{-1}\left(\Phi^{-1}\left(P_i + \frac{C}{\sqrt{n}}\right)\right) = G_n^+(P_i) \; . \tag{3.8}$$

for all i by applying $\zeta^{-1} \circ \Phi^{-1}$ to both sides and substituting $F_n(i) = P_i \in I_n$. Note that, by the definition of G_n^-, G_n^+, F_n^{-1} and Equation (3.8) we also have

$$G_n^-(P_\ell) = G_n^-(P_{\ell+1}) - 1 \leq \ell = F^{-1}(P_{\ell+1}) - 1 = F^{-1}(P_\ell)$$

and

$$G_n^+(P_\ell) = G_n^+(P_{\ell+1}) - 1 \geq \ell = F^{-1}(P_{\ell+1}) - 1 = F^{-1}(P_\ell) \; .$$

Similarly, we have

$$G_n^-(P_r) = G_n^-(P_{r-1}) + 1 \leq r = F^{-1}(P_{r-1}) + 1 = F^{-1}(P_r)$$

and

$$G_n^+(P_r) = G_n^+(P_{r-1}) + 1 \geq r = F^{-1}(P_{r-1}) + 1 = F^{-1}(P_r) \; .$$

Hence we have

$$G_n^-(P_i) \leq i \leq G_n^+(P_i)$$

for all $P_i \in \mathcal{P}_n := \{P_i \in I_n\} \cup \{P_\ell, P_r\}$. Substituting $i = F_n^{-1}(P_i)$ for $P_i = F_n(i) \in I_n$ we obtain

$$G_n^-(P_i) \leq F^{-1}(P_i) \leq G_n^+(P_i) \tag{3.9}$$

for all $P_i \in \mathcal{P}_n$ due to the definition of the inverse F_n^{-1} and Equation (3.3). That is, G_n^- is a minorant for F_n^{-1} and G_n^+ is a majorant of F_n^{-1} for all $P_i \in \mathcal{P}_n$. Next, we show that G_n^- is a minorant of F_n^{-1} and $G_n^+ + 1$ is a majorant of F_n^{-1} for all $p \in I_n$. Consider an index i such that $(P_{i-1}, P_i) \cap I_n \neq \emptyset$, that is $P_{i-1}, P_i \in \mathcal{P}_n$, and let $p \in (P_{i-1}, P_i) \cap I_n$. Then $F^{-1}(p) = i$ and we have

$$\begin{aligned}
G_n^-(p) &\leq G_n^-(P_i) && \text{since } G_n^- \text{ is non-decreasing}\\
&\leq F^{-1}(P_i) && \text{due to Equation (3.9)}\\
&= i\\
&= F^{-1}(p)\ .
\end{aligned}$$

Similarly, we have

$$\begin{aligned}
G_n^+(p) + 1 &\geq G_n^+(P_{i-1}) + 1 && \text{since } G_n^+ \text{ is non-decreasing}\\
&\geq F^{-1}(P_{i-1}) + 1 && \text{due to Equation (3.9)}\\
&= i\\
&= F^{-1}(p)\ .
\end{aligned}$$

Therefore, G_n^- is a minorant for F^{-1} for all $p \in I_n$ and $G_n^+ + 1$ is a majorant for F^{-1} for all $p \in I_n$. Since both F_n^{-1} and G_n are bounded by G_n^- and $G_n^+ + 1$ from above and below, respectively, on I_n, the absolute difference of these functions must be bounded by

$$\left| F_n^{-1}(p) - \zeta^{-1}(\Phi(p)) \right| \leq G_n^+(p) - G_n^-(p) + 1$$

on I_n. Note that, since $G_n^+(p) - G_n^-(p) = G_n^+(P_{\ell+1}) - G_n^-(P_{\ell+1})$ for all $p \in (0, P_{\ell+1})$ and $G_n^+(p) - G_n^-(p) = G_n^+(P_{r-1}) - G_n^-(P_{r-1})$ for all $p \in (P_{r-1}, 1)$, the supremum of this difference is attained in the interval $[P_{\ell+1}, P_{r-1}]$. Using this observation along with the definition of ζ^{-1} we get

$$\begin{aligned}
\lim_{n\to\infty} \sup_{p\in I_n} \left\{ G_n^+(p) - G_n^-(p) + 1 \right\} &= \lim_{n\to\infty} \sup_{p\in I_n} \frac{\sqrt{n}}{2} \left\{ \Phi^{-1}\left(p + \frac{C}{\sqrt{n}}\right) - \Phi^{-1}\left(p - \frac{C}{\sqrt{n}}\right) \right\} + 1\\
&= \lim_{n\to\infty} \sup_{p\in I_n} \frac{2C}{\sqrt{n}} \cdot \frac{\sqrt{n}}{2} \cdot \frac{\Phi^{-1}\left(p + \frac{C}{\sqrt{n}}\right) - \Phi^{-1}\left(p - \frac{C}{\sqrt{n}}\right)}{\frac{2C}{\sqrt{n}}} + 1\ .
\end{aligned}$$

Since the third term of the previous equation is equal to the difference quotient of Φ^{-1} at p, this simplifies to

$$\begin{aligned}
\lim_{n\to\infty} \sup_{p\in I_n} \left\{ G_n^+(p) - G_n^-(p) + 1 \right\} &= \sup_{p\in I_n} C \cdot \frac{\partial}{\partial p} \Phi^{-1}(p) + 1\\
&= \sup_{p\in I_n} C \cdot \left\{ \frac{1}{\sqrt{2\pi}} \cdot \exp\left(-\frac{\Phi^{-1}(p)^2}{2} \right) \right\}^{-1} + 1
\end{aligned}$$

by the definition of the derivative of Φ^{-1}, which yields

$$\lim_{n\to\infty} \sup_{p\in I_n} \left\{ G_n^+(p) - G_n^-(p) + 1 \right\} = \sup_{p\in I_n} \sqrt{2\pi} \cdot C \cdot \exp\left\{ \frac{\Phi^{-1}(p)^2}{2} \right\} + 1 \, .$$

Note that by the symmetry of Φ we have $\Phi^{-1}(p) = -\Phi^{-1}(1-p)$ and, therefore $\Phi^{-1}(p)^2 = \Phi^{-1}(1-p)^2$. Further, note that $\Phi^{-1}(p)^2$ is minimized for $p = 0.5$ and monotonically increasing for $p \to 0$. This yields

$$\lim_{n\to\infty} \sup_{p\in I_n} \left\{ G_n^+(p) - G_n^-(p) + 1 \right\} \leq \sqrt{2\pi} \cdot C \cdot \exp\left\{ \frac{\Phi^{-1}(p_0)^2}{2} \right\} + 1 \, ,$$

which concludes the proof since $I \subset I_n$. □

Although the rate of deterioration of the error bound provided in the previous lemma is rather fast towards the boundaries of the domain, it yields that the absolute error of the approximation is less than 8 for $0.05 \leq p \leq 0.95$ in the limit, which shows that the approximation can be quite useful already. In fact this means that the probability of witnessing an error larger than 8 while generating approximately binomially distributed random variates using the inverse normal distribution is less than 0.1 for large values of n. Figure 3.3 shows the quality of the bound provided by the minorants and majorants for large n on a log-scale plot as well as the bound provided by the lemma. Note that the error is less than 1 for a large fraction of the possible probabilities p, while the image of the function we approximate is $\{1, \ldots, n\}$.

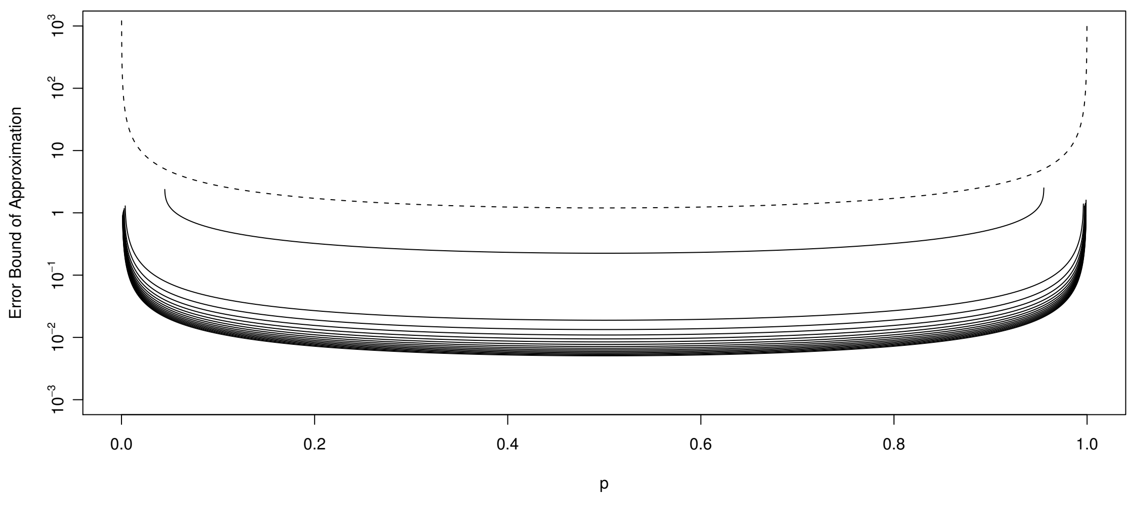


Figure 3.3: Log-scale plot of the absolute error bound of the approximation to the inverse of the binomial distribution function given by $G_n^+(p) - G_n^-(p)$ for $p \in I_n$ for different values of in $n \in [500, 10^6]$ (solid lines) and $\sqrt{2\pi} \cdot C \cdot \exp\left\{\frac{\Phi^{-1}(p)^2}{2}\right\}$ (dashed line), respectively.

Next, we approximate the normal distribution function and its inverse numerically in order to be able to evaluate these functions efficiently. The standard normal distribution function Φ can be approximated according to Equation 26.2.19 in Abramowitz and Stegun [AS72]. Let

$$\begin{array}{lll} d_1 = 0.0498673470 & d_2 = 0.0211410061 & d_3 = 0.0032776263 \\ d_4 = 0.0000380036 & d_5 = 0.0000488906 & d_6 = 0.0000053830\,. \end{array}$$

Then $\Phi(x)$ can be approximated by

$$G(x) = 1 - \frac{1}{2}\left(1 + \sum_{i=1}^{6} d_i x^i\right)^{-16} \tag{3.10}$$

with error $|\varepsilon_G(x)| \leq 1.5 \cdot 10^{-7}$ for $0 \leq x$. Further, we can approximate the inverse of the standard normal distribution using Padé approximants [PTVF07]. For instance, let

$$\begin{array}{ll} a_1 = 4.442882938158366 & a_2 = 5.38865628256879 \\ a_3 = 10.882423209201171 & a_4 = 6.614276302275924 \\ a_5 = 8.20719310378084 & a_7 = 1.923455149786169 \\ a_9 = 0.7895597958079597 & \end{array}.$$

Then the Padé approximant for Φ^{-1} in $x = 0.5$ computed by Mathematica [Inc08] is given by

$$H(x) = \frac{a_1 t - a_3 t^3 + a_5 t^5 + a_7 t^7 + a_9 t^9}{1 - a_2 t^2 + a_4 t^4}\,, \tag{3.11}$$

where $t = x - 0.5$. We have $|H(x) - Q(x)| \leq 3.75172 \cdot 10^{-7}$ for all $x \in [0.2, 0.8]$, where $Q(x)$ is the approximation of the inverse cumulative distribution function of the standard normal distribution built into Mathematica. In order to extend the range of the approximation, we can compute additional Padé approximants in the sub-ranges of $[0, 0.2)$ and $(0.8, 1]$, respectively. Note that these functions can be computed using only a constant number of additions, multiplications and divisions, that is they can be computed in constant time in the RAM machine model.

Next, we show that the additional absolute error resulting from approximating the inverse of the standard normal distribution function numerically does not increase by too much when using reasonable numerical approximations. While this is immediately obvious for our approximation to the binomial distribution where the operand is transformed prior to the evaluation of the standard normal distribution function, it is not immediately clear for our approximation of the inverse of the binomial distribution since this approximation involves a transformation of the function value of the numerical approximation.

Lemma 3.6. *Let $n \in \mathbb{N}$ and let $0 \leq k \leq n$. Let $(H_n)_{n\geq 0}$ be a family of approximations of the inverse of the standard normal distribution function with absolute error bounded by $\varepsilon_n \in o(n^{-1/2})$. Further, let F_n be the distribution function of the binomial distribution $\mathrm{Binom}(n, \frac{1}{2})$ and let U be a random uniform $[0, p]$-variate such that $p = F_n(k)$. Let F_n^{-1} denote the inverse of F_n as defined in Equation (3.2). Further, let $P_i := F_n(i)$ for $0 \leq i \leq n$ and let $I = [p_0, 1 - p_0]$ for a fixed value $0 < p_0 < \frac{1}{2}$ such that $p_0 > \frac{C}{\sqrt{n}}$. Let $\ell := \max\{i \mid P_i < p_0\}$*

and $r := \min\{i \mid P_i > 1 - p_0\}$ *and let* G_n *be defined as in Lemma 3.5. Let*

$$\widetilde{G}_n(p) := \begin{cases} \widetilde{G}_n(P_{\ell+1}) - 1 & p \in (0, P_{\ell+1}) \\ \lceil \zeta^{-1}(H_n(p)) \rceil & p \in [P_{\ell+1}, P_{r-1}] \\ \widetilde{G}_n(P_{r-1}) + 1 & p \in (P_{r-1}, 1) \end{cases}$$

Then $\widetilde{G}_n(U)$ *approximates the* k*-restricted binomial distribution with absolute error*

$$\lim_{n\to\infty} \sup_{U \in I} \left| \widetilde{G}_n(U) - F_n^{-1}(U) \right| \le \sqrt{2\pi} \cdot C \cdot \exp\left\{ \frac{\Phi\left(p_0\right)^2}{2} \right\} + 2$$

on the interval I.

Proof. Note that F_n^{-1} is constant on $(P_\ell, P_{\ell+1}]$ and $(P_{r-1}, P_r]$ and note that both these intervals are overlapping I. Hence, the maximum absolute error of the function $|\widetilde{G}_n(U) - F_n^{-1}(U)|$ is attained in the interval $[P_{\ell+1}, P_{r-1}]$. In this interval, we have

$$\lim_{n\to\infty} \sup_{U \in I} \left| \widetilde{G}_n(U) - F_n^{-1}(U) \right| = \lim_{n\to\infty} \sup_{U \in I} \left| \widetilde{G}_n(U) - \zeta^{-1}(\Phi^{-1}(U)) \right| + \sqrt{2\pi} \cdot C \cdot \exp\left\{ \frac{\Phi^{-1}\left(p_0\right)^2}{2} \right\} + 1$$

using Lemma 3.5

$$\le \lim_{n\to\infty} \sup_{U \in I} \left| \zeta^{-1}(H_n(U)) - \zeta^{-1}(\Phi^{-1}(U)) \right| + \sqrt{2\pi} \cdot C \cdot \exp\left\{ \frac{\Phi^{-1}\left(p_0\right)^2}{2} \right\} + 2$$

using the definition of $\widetilde{F}_n^{-1}$ and the fact that rounding causes at most an additive unit error

$$\le \lim_{n\to\infty} \sup_{U \in I} \left| \zeta^{-1}(\Phi^{-1}(p_0) \pm \varepsilon_H) - \zeta^{-1}(\Phi^{-1}(U)) \right| + \sqrt{2\pi} \cdot C \cdot \exp\left\{ \frac{\Phi^{-1}\left(p_0\right)^2}{2} \right\} + 2$$

by the quality of the approximation H_n

$$\le \lim_{n\to\infty} \varepsilon_n \frac{\sqrt{n}}{2} + \sqrt{2\pi} \cdot C \cdot \exp\left\{ \frac{\Phi^{-1}\left(p_0\right)^2}{2} \right\} + 2 \,,$$

by the definition of ζ^{-1}, which by $\varepsilon_n \in o(n^{-1/2})$ yields

$$= \sqrt{2\pi} \cdot C \cdot \exp\left\{ \frac{\Phi^{-1}\left(p_0\right)^2}{2} \right\} + 2 \,.$$

□

Algorithm 3.3: Approximately k-restricted binomial random variate generation

```
Input: $n$, $k$
Output: approximately $k$-restricted binomial random Variate $X$
1  $p \leftarrow \widetilde{F}_n(k)$ where $\widetilde{F}_n(k)$ is defined as in Lemma 3.4
2  $U \leftarrow$ random uniform $[0,1]$-variate
3  $U' \leftarrow Up$                    // generate random uniform $[0,p]$-variate
4  $X \leftarrow \widetilde{G}_n(U')$ where $\widetilde{G}_n$ is defined as in Lemma 3.6
5  if $X < 0$ then
6      return 0
7  else if $X > k$ then
8      return $k$
9  else
10     return $X$
```

That is, in order to keep the additional absolute error introduced by the numerical approximation to the inverse of the normal distribution low, it suffices to approximate the inverse of the standard normal distribution with error bounded by $o(n^{-1/2})$. Using an approximation of the normal quantile function with absolute error bounded by $\varepsilon_H \leq 10^{-7}$, for instance, the additional absolute error introduced by the approximation of the normal quantile function, given by $\varepsilon_H\sqrt{n}/2 + 1$, is less than 2 for all $n < 4 \cdot 10^{14}$. In Equation (3.11) we showed that these error terms are indeed within reach of reasonable approximations.

To summarize, we propose the following algorithm for generating approximately k-restricted binomial random variates, as illustrated in Algorithm 3.3. The algorithm first computes $p \approx F_n(k)$ using a numerical approximation of the standard normal distribution such as the one in Equation (3.10). This is justified by Lemma 3.4. Then it generates a random uniform $[0,p]$-variate U' using a simple transformation from a random uniform $[0,1]$ variate U. Finally, we apply a numerical approximation of the standard normal quantile function such as the one in Equation (3.11) to U and apply ζ^{-1} according to Lemma 3.3 to transform U' into an approximately k-restricted binomial random variate. Due to the error introduced in the computation, X may be larger than k or smaller than 0. By substituting these values with 0 and k, respectively, we obtain an approximately k-restricted binomial random variate.

The quality of the resulting approximation is better for large n and if k is not too small compared to n. For large n, and if k is large compared to n, the approximation guarantees discussed numerically suggest that the approximation is very close to the k-restricted binomial distribution. The quality of the approximation can be improved at the cost of more complicated Padé approximants.

Theorem 3.3. *Algorithm 3.3 generates an approximately k-restricted binomial random variate in time $\mathcal{O}(1)$.*

Proof. The quality of the approximation of Algorithm 3.3 mainly depends on Lemma 3.6 and the used numerical approximations for the standard normal distribution and has been discussed above. We assume that the numerical approximations for the standard normal distribution can be computed by a constant number of numerical operations involving only addition, multiplication and division. As an example, consider the approximations given in Equations (3.10) and (3.11). Since the numerical approximations can be computed with a constant number of operations, the algorithm can be implemented to run im $\mathcal{O}(1)$ time. □

3.5 Generating Well-Ordered Degenerate Graphs

In this section we study the problem of generating well-ordered k-degenerate graphs with n vertices or n vertices and m edges uniformly at random. We first establish recursive formulae for the cardinalities of the respective graph classes. Then we show how these formulae can be used to generate graphs from these classes uniformly at random. Since the involved probability distributions are hard to deal with computationally, we additionally present non-uniform and easy-to-implement generators, whose running time is optimal.

Lemma 3.7. *We have the following recursive formulae for the cardinalities $D(n,m,k)$ of $\mathcal{D}(n,m,k)$ and $D(n,k)$ of $\mathcal{D}(n,k)$*

$$D(n,m,k) = \sum_{i=0}^{\min\{n-1,m,k\}} \binom{n-1}{i} D(n-1,m-i,k) \tag{3.12}$$

$$D(n,k) = \sum_{i=0}^{\min\{n-1,k\}} \binom{n-1}{i} D(n-1,k) \; . \tag{3.13}$$

Proof. Let G be a well-ordered k-degenerate graph with n vertices and m edges. Then the graph $G[v_2,\ldots,v_n]$ is a well-ordered k-degenerate graph with $n-1$ vertices and m' edges, where $m' = m-d$ for some $0 \le d \le \min\{n-1,m,k\}$. To see this, note that the degree of v_1 is at most $\min\{n-1,m,k\}$ since G is well-ordered k-degenerate and that removing v_1 and its adjacent edges does not affect labels and the out-degree of the remaining vertices. On the other hand, suppose that G' is a well-ordered k-degenerate graph such that the minimum vertex-label of G' is $i+1$ for $i > 1$ and suppose that G' has n' vertices and m' edges. Let G be the graph resulting from adding a new vertex v_i labeled i and a set of at most $\min\{n-1,m,k\}$ edges incident to v_i. Then, clearly, G is well-ordered k-degenerate, since adding the edges does not affect the labeling and the out-degree of the other vertices and since the out-degree of v_i is at most $\min\{n-1,m,k\}$.

That is, for fixed d such that $0 \le d \le \min\{n-1,m,k\}$ we obtain all k-degenerate graphs with n vertices and m edges as follows. We take an arbitrary well-ordered k-degenerate graph G' with $n-1$ vertices $\{v_1,\ldots,v_{n-1}\}$ and $m' = m-d$ edges and create a new well-ordered k-degenerated graph with n vertices and m edges by first re-labeling the vertices of G' such that the new label v_i is $i+1$, adding a new vertex labeled 1 and adding $d \le \min\{n-1,m,k\}$ edges to the resulting graph. Since the newly added vertex chooses d neighbors out of $n-1$ vertices we obtain

$$D(n,m,k) = \sum_{d=0}^{\min\{n-1,m,k\}} \binom{n-1}{d} D(n-1,m-d,k) \; .$$

Clearly, each of these graphs is unique and no graph is counted more than once. Similarly, if G is a k-degenerate graph with n vertices, then $G[v_2,\ldots,v_n]$ is k-degenerate with $n-1$ vertices.

Algorithm 3.4: DEGENERATE(i, n, k)

```
Input: $n \in \mathbb{N}$, $0 \le k \le n$, $1 \le i \le n$
Output: random well-ordered $k$-degenerate graph with $n - i + 1$ vertices $v_i, \dots, v_n$
1 if $i = n$ then
2     return $(\{v_n\}, \emptyset)$
3 $d_i \leftarrow \text{Binom}_{\le \min\{n-i,k\}}(n-i)$                    $X_i$
4 $(V', E') \leftarrow$ DEGENERATE$(i+1, n, k)$                     $Y_{i+1}$
5 $S \leftarrow \text{Uniform}\left(\binom{V'}{d_i}\right)$                          $Z_i$
6 $E \leftarrow E' \cup \{\{v_i, s\} \mid s \in S\}$
7 $G \leftarrow (V \cup \{v_i\}, E)$
8 return $G$                                                $Y_i$
```

We can construct all k-degenerate graphs with n vertices from the k-degenerate graphs with $n-1$ vertices by adding a new vertex as before and choosing $0 \le d \le \min\{n-1, k\}$ edges incident to the newly added vertex. Hence, we have

$$D(n,k) = \sum_{d=0}^{\min\{n-1,k\}} \binom{n-1}{d} D(n-1,k) \quad .$$

□

The uniform samplers presented in the following sections are based on these recursive formulae.

3.5.1 Generating k-degenerate Graphs with n Vertices

First, we describe an exact uniform sampler for k-degenerate graphs with a given number of vertices. Our approach is a recursive algorithm that expects an index i as well as the number of vertices n of the graph as parameters. If $i = n$ the algorithm returns a graph consisting of an isolated vertex with label n. Otherwise, it randomly chooses the successor-degree d_i of vertex v_i according to a k-restricted binomial distribution (see Table 3.2) and recursively calls itself generating a well-ordered k-degenerate graph G' with $n - i$ vertices labeled $v_{i+1}, \dots, v_n$. Then it chooses d_i distinct vertices uniformly at random, creates a new vertex labeled i and inserts new edges from v_i to all d_i selected vertices. The resulting graph G is returned. The details are listed in Algorithm 3.4. It remains to show that Algorithm 3.4 samples well-ordered k-degenerate graphs with n vertices uniformly at random if invoked with DEGENERATE$(1, n, k)$. Let Y_1 denote the random variable representing the outcome of the call DEGENERATE$(1, n, k)$.

Theorem 3.4. *Algorithm 3.4 generates all graphs in $\mathcal{D}(n,k)$ with equal probability, that is, $Y_1 \sim \text{Uniform}(\mathcal{D}(n,k))$ if invoked with* DEGENERATE$(1, n, k)$. *The algorithm can be implemented to run in $\mathcal{O}(n \log k + m)$ time in the real-RAM machine model.*

Proof. First, we show that the graphs generated by Algorithm 3.4 are well-ordered k-degenerate if the algorithm is invoked with DEGENERATE$(1, n, k)$. This follows from the invariant of the algorithm that the graph generated by the recursive call to the algorithm in Line 4 is a well-ordered k-degenerate graph with $n-i$ vertices $v_{i+1},\dots,v_n$. We show that the invariant is maintained by induction on i. The invariant is certainly true for $i=n$. Suppose that the invariant is true for $i+1$. Then the call to DEGENERATE(i,n,k) recursively calls DEGENERATE$(i+1,n,k)$, which produces a well-ordered k-degenerate graph G' with $n-i$ vertices $v_{i+1},\dots,v_n$. By adding a vertex v_i with at most $\min\{k,n-i\}$ edges according to Line 6 we obtain a well-ordered k-degenerate graph with $n-i+1$ vertices $v_1,\dots,v_n$.

Next, we show that each well-ordered k-degenerate graph G with n vertices $v_1,\dots,v_n$ is generated with probability $D(n,k)^{-1}$. Let $G_i := G[v_i,\dots,v_n]$. Then G_i is a well-ordered k-degenerate graph with $n-i+1$ vertices $v_i,\dots,v_n$. Let X_i, Y_{i+1}, Z_i denote the random variables representing the outcome of the randomized operations in Lines 3, 4 and 5 of the algorithm, respectively. That is, $X_i = d_i$, $Y_i = G_i$, $Y_{i+1} = G_{i+1}$ and $Z_i = S$. We prove that $Y_i \sim \text{Uniform}(\mathcal{D}(n-i+1,k))$ by induction on the parameter i of the algorithm. Since there is only one well-ordered k-degenerate graph with one vertex we have $D(1,k)=1$ and, thus, the claim holds for $i=n$. Assume that $i<n$. Let $G_i = (\{v_i,\dots v_n\},E)$ be a well-ordered k-degenerate graph and let d_i denote the degree of v_i. Further, let $N(v_i) \subseteq \{v_{i+1},\dots,v_n\}$ denote the neighbors of v_i. According to the definition of the algorithm the probability of obtaining G_i given as input i and n is given by

$$\mathbb{P}(Y_i = G_i) = \mathbb{P}(X_i = d_i) \cdot \mathbb{P}(Y_{i+1} = G_i - v_i) \cdot \mathbb{P}(Z_i = N(v_i) \mid X_i = d_i) .$$

By definition of the k-restricted binomial distribution, we have

$$\mathbb{P}(X_i = d_i) = \frac{\binom{n-i}{d_i}}{\sum_{j=0}^{\min\{k,n-i\}} \binom{n-i}{j}}$$

and by choosing d_i neighbors uniformly at random, we have

$$\mathbb{P}(Z_i = N(v_i) \mid X_i = d_i) = \frac{1}{\binom{n-i}{d_i}} .$$

Thus, by induction hypothesis $Y_{i+1} \sim \text{Uniform}\left(\mathcal{D}(n-i,k)\right)$ we obtain

$$\begin{aligned}\mathbb{P}(Y_i = G_i) &= \frac{\binom{n-i}{d_i}}{\sum_{j=0}^{\min\{k,n-i\}} \binom{n-i}{j}} \cdot \frac{1}{D(n-i,k)} \cdot \frac{1}{\binom{n-i}{d_i}} \\ &= \frac{1}{\sum_{j=0}^{\min\{k,n-i\}} \binom{n-i}{j} D(n-i,k)} ,\end{aligned}$$

which simplifies to

$$\mathbb{P}(Y_i = G_i) = \frac{1}{D(n-i+1,k)}$$

using Equation (3.13). Hence, $\mathbb{P}(Y_1 = G) = D(n,k)^{-1}$ as claimed.

By Lemma 3.2 we can generate X_i in Line 3 in $\mathcal{O}(\log k)$ time using values, which have been pre-computed in $\mathcal{O}(n)$ time and space. Since Line 3 is performed for each of the $n-1$

vertices $v_1, \dots, v_{n-1}$ of the graph and since the pre-computation must only be performed once, the pre-computation is completely amortized. Hence Line 3 can be implemented to run in amortized $\mathcal{O}(\log k)$ time. Further, we can implement Line 5 iteratively as follows. Suppose that V' is implemented as an array with random access. When picking the j-th element for $j \leq d_i$ we choose a random index r between 1 and $|V'| - j + 1$ uniformly at random. Then we swap the elements r and $|V'| - j + 1$. Hence, no element is chosen more than once. The probability obtaining a specific set $N(v_i)$ is given by

$$d_i! \cdot \frac{1}{|V'|} \cdot \frac{1}{|V'| - 1} \cdots \frac{1}{|V'| - d_i + 1} = \binom{|V'|}{d_i}^{-1} .$$

To choose a random index between 1 and $t > 1$ uniformly at random we first compute a random uniform $[0, 1]$-variate U and compute $X = 1 + \lfloor (t-1)U \rfloor$. Then

$$\begin{aligned}\mathbb{P}(X = j) &= \mathbb{P}\left(\lfloor (t-1)U \rfloor = j - 1\right) \\ &= \mathbb{P}(j - 1 \leq (t-1)U < j + 1) \\ &= \mathbb{P}\left(\frac{j-1}{t-1} \leq U < \frac{j}{t-1}\right) \\ &= \frac{j - (j-1)}{t-1} = \frac{1}{t-1} ,\end{aligned}$$

that is, each $j \in \{1, \dots, t\}$ is chosen with equal probability. Note that the floor operation is implemented in hardware on many modern floating-point units and is therefore considered as a constant-time operation here. Then the number of calls in Line 5 is proportional to the number of generated edges. If the generated graph has m edges, then the resulting time complexity is $\mathcal{O}(n \log k + m)$. □

In Section 3.4 we have seen how to generate random variates with an approximately k-restricted binomial distribution in expected or deterministic $\mathcal{O}(1)$ time using the inversion method and approximations for the normal and inverse normal distributions. Thus, by substituting the exact random variate generator for in Algorithm 3.4 by Algorithm 3.3 we obtain an approximately uniform generator for k-degenerate graphs with optimal running time as summarized in the following theorem.

Theorem 3.5. *Using Algorithm 3.3 to sample random variates with an approximately k-restricted binomial distribution Algorithm 3.4 can be turned into an approximate uniform generator for $\mathcal{D}(n, k)$ with running time $\mathcal{O}(n + m)$ in the standard RAM machine model.*

3.5.2 Generating k-degenerate Graphs with n Vertices and m Edges

Next, we present a uniform generator for well-ordered k-degenerate graphs with n vertices and m edges. Throughout this section we assume $m \leq M(n, k)$, where $M(n, k)$ is the maximum number of edges of a well-ordered k-degenerate graph with n vertices. Since the degree of each vertex v_i of a well-ordered k-degenerate graph with n vertices $v_1, \dots, v_n$ is bounded by $\min\{k, n - i\}$ we have

$$M(n, k) := \sum_{i=1}^{n} \min\{k, n - i\} .$$

Algorithm 3.5: Degenerate(i, n, m, k)

```
Input: $n \in \mathbb{N}$, $0 \le k \le n$, $1 \le i \le n$, $0 \le m \le M(n-i+1, k)$,
Output: random well-ordered $k$-degenerate graph with $n-i+1$ vertices $v_i, \dots, v_n$
        and $m$ edges
1 if $m = 0$ then
2     return $(\{v_i, \dots, v_n\}, \emptyset)$
3 $d_i \leftarrow$ choose $0 \le d_i \le \min\{n-i, m, k\}$ proportional to $\binom{n-i}{d_i} D(n-i, m-d_i, k)$    $X_i$
4 $(V, E') \leftarrow$ Degenerate$(i+1, n-i, m-d_i, k)$    $Y_{i+1}$
5 $S \leftarrow$ Uniform$\left(\binom{V'}{d_i}\right)$    $Z_i$
6 $E \leftarrow E' \cup \{(v_i, s) \mid s \in S\}$
7 $G \leftarrow (V \cup \{v_i\}, E)$
8 return $G$    $Y_i$
```

For $n \ge k$ this yields

$$M(n,k) = nk - \sum_{i=1}^{k} i = nk - \frac{k(k+1)}{2} = nk - \binom{k+1}{2} . \tag{3.14}$$

For $n < k$ this yields

$$M(n,k) = \sum_{i=1}^{n} n - i = \sum_{i=0}^{n-1} i = \frac{n(n-1)}{2} = \binom{n}{2} .$$

Clearly, there exist well-ordered k-degenerate graphs with n vertices and $M(n,k)$ edges. In order to see this, it suffices note that, for each vertex v_i there are at least $\min\{k, n-i\}$ vertices v_j with $j > i$. Hence, we can construct a graph such that the degree of v_i is exactly $\min\{k, n-i\}$ for all $1 \le i \le n$. On the other hand, however, no k-degenerate graph with n vertices can have more than $M(n,k)$ edges since this number of edges can only be achieved if each vertex has the maximum number of edges it is allowed to have. Finally, note that the empty graph on vertex set $\{v_1, \dots, v_n\}$ is well-ordered k-degenerate, that is, the minimum number of edges of a well-ordered k-degenerate graph equals zero.

The algorithm we propose is similar to the algorithm presented in the previous section. At first we choose $0 \le d_i \le \min\{n-1, m, k\}$ proportional to $\binom{n-1}{d_i} D(n-1, m-i, k)$. Then we recursively choose G' uniformly at random from $\mathcal{D}(n-1, m-i, k)$. Next we choose d_i vertices uniformly at random from G' and finally we create a new vertex that is connected to all selected vertices in G'. The pseudo-code is listed in Algorithm 3.5 and its correctness is established in the following theorem.

Theorem 3.6. *Algorithm 3.5 generates all graphs in $\mathcal{D}(n,m,k)$ with equal probability, that is, $Y_1 \sim$ Uniform$(\mathcal{D}(n,m,k))$, if invoked with* Degenerate$(1, n, m, k)$. *The algorithm can be implemented to run in $\mathcal{O}(n \log k + m + nk)$ time in the real-RAM machine model.*

Proof. The proof is analogous to the proof of Theorem 3.4. First, we show that the graphs generated by calling Degenerate$(1, n, m, k)$ are well-ordered k-degenerate with n vertices and m edges. This follows from the invariant that a call to Degenerate(i, n, m', k)

with $m' \leq \min\{M(n,k), m\}$ yields a well-ordered k-degenerate graph with $n-i+1$ vertices $v_i, \dots, v_n$ and m' edges. To see, why the invariant is maintained, note that a call to Degenerate$(i,n,0,k)$ yields a well-ordered k-degenerate graph with $n-i+1$ vertices $v_i, \dots, v_n$ as claimed for all $1 \leq i \leq n$. Hence, the invariant holds for these calls. Consider a call to Degenerate(i,n,m,k) and suppose that the invariant is maintained for all $j > i$ and $m' \leq m$ such that $D(j,m',k) \neq \emptyset$. Then the recursive call in Line 4 yields a well-ordered k-degenerate graph G_{i+1} with $n-i$ vertices $v_{i+1}, \dots, v_n$ and $m-d_i$ edges such that $0 \leq d_i \leq \min\{k, n-i\}$. By adding a new vertex v_i and d_i edges incident to v_i, we obtain a new graph G_i with $n-i+1$ vertices $v_i, \dots, v_n$ and $m'+d_i$ edges. Hence, the invariant is maintained.

Next, we show that each graph $G \in \mathcal{D}(n,m,k)$ is generated with probability $D(n,m,k)^{-1}$. By induction over i and m' we show that $Y_i \sim \text{Uniform}\left(D(i,m',k)\right)$. As above, the induction hypothesis holds for all $1 \leq i \leq n$ and $m' = 0$ since there is only one well-ordered k-degenerate graph with $n-i+1$ vertices $v_i, \dots, v_n$. Assume that the induction hypothesis holds for all $j < i$ and all $m' \leq m$ such that $\mathcal{D}(j,m',k) \neq \emptyset$ and consider the call Degenerate(i,n,m,k). We again let X_i, Y_{i+1}, Z_i denote the random variables for the outcome of the randomized operations in Lines 3, 4 and 5 of the algorithm, respectively. The probability of obtaining a graph $G_i \in \mathcal{D}(n-i+1,m,k)$ with $n-i+1$ vertices $v_i, \dots, v_n$ and m edges from a call to Degenerate(i,n,m,k) such that the degree of v_i equals d_i is given by

$$\mathbb{P}(Y_i = G_i) = \mathbb{P}(X_i = d_i) \cdot \mathbb{P}(Y_{i+1} = G_i - v_i) \cdot \mathbb{P}(Z_i = N(v_i) \mid X_i = d_i)\ .$$

By choosing d_i proportional to $\binom{n-i}{d_i} D(n-i, m-d_i, k)$ we have

$$\mathbb{P}(X_i = d_i) = \frac{\binom{n-i}{d_i} D(n-i, m-d_i, k)}{\sum_{j=0}^{\min\{n-i,m,k\}} \binom{n-i}{j} D(n-i, m-j, k)}$$

and by choosing d_i neighbors uniformly at random, we have

$$\mathbb{P}(Z_i = N(v_i) \mid X_i = d_i) = \frac{1}{\binom{n-i}{d_i}}\ .$$

Thus, by induction hypothesis $Y_{i+1} \sim \text{Uniform}\left(\mathcal{D}(n-i, m-d_i, k)\right)$ we obtain

$$\mathbb{P}(Y_i = G_i) = \frac{1}{\binom{n-i}{d_i}} \cdot \frac{1}{D(n-i, m-d_i, k)} \cdot \frac{\binom{n-i}{d_i} Dn-i, m-d_i, k)}{\sum_{j=0}^{\min\{n-i,m,k\}} \binom{n-i}{j} D(n-i, m-j, k)}\ ,$$

which simplifies to

$$\mathbb{P}(Y_i = G_i) = \frac{1}{D(n-i+1, m, k)}$$

using Equation (3.12). Hence, by induction we generate each graph in $\mathcal{D}(n,m,k)$ with probability $D(n,m,k)^{-1}$ by calling Degenerate$(1,n,m,k)$.

The algorithm can be implemented with similar techniques as those described in the proof of Theorem 3.4. Line 3 can be implemented using the discrete inversion method described

Algorithm 3.6: DEGENERATE(n, m, k)

```
Input: $n \in \mathbb{N}$, $0 \leq k \leq n$, $0 \leq m \leq M(n,k)$
Output: well-ordered $k$-degenerate graph $G = (V, E)$ with $n$ vertices and $m$ edges
1  $V \leftarrow \{v_1, \dots, v_n\}$
2  $E \leftarrow \emptyset$
3  $d_i \leftarrow 0$  for all $1 \leq i \leq n$
4  $C \leftarrow \{v_i \in V \mid \min\{n-i, k\} - 1 \geq 0\}$
5  for $i = 1$ to $m$ do
6      $v_i \leftarrow \text{Uniform}(C)$
7      $d_i \leftarrow d_i + 1$
8      if $d_i = \min\{n-i, k\}$ then
9          $C \leftarrow C \setminus \{v_i\}$
10 for $i = n-1$ to $1$ do
11     $X \leftarrow \text{Uniform}\left(\binom{\{v_{i+1}, \dots, v_n\}}{d_i}\right)$
12     $E \leftarrow E \cup \{(v_i, x) \mid x \in X\}$
13 return $G := (V, E)$
```

in Lemma 3.1. For each i we sample the degrees d_i from a different distribution. Each of these distributions is defined over most k different values. Hence, by Lemma 3.1 we can sample the values in time $\mathcal{O}(\log k)$ if we allow for a total pre-computation of $\mathcal{O}(nk)$ time and space, respectively. Since the returned graph has m edges the resulting running time is $\mathcal{O}(n \log k + m + nk)$. □

Note that, since $m \in \mathcal{O}(nk)$, the algorithm is rather efficient for large values of m, whereas the overhead of the precomputation dominates for small values of m.

Next, we propose an efficient complete non-uniform generator for well-ordered k-degenerate graphs with n vertices and m edges. The algorithm first chooses a successor-degree sequence $d_1, \dots, d_n$ and then chooses d_i vertices uniformly at random for each vertex. In order to choose the successor-degree sequence $d_1, \dots, d_n$ we maintain a set C containing the vertices whose tentative degree is at most $\min\{n-i, k\} - 1$. Then we iteratively choose a random vertex from C and augment its degree. If the degree of a vertex reaches $\min\{n-i, k\}$ it is removed from C. In order to create the edges according to the chosen degree sequence, the algorithm then iterates over the vertices starting at the vertex with index $n-1$ and chooses d_i target vertices with higher index where d_i denotes the successor-degree of vertex v_i. The pseudo-code can be found in Algorithm 3.6.

Theorem 3.7. *Algorithm 3.6 can be implemented to run in time $\mathcal{O}(n+m)$ in the RAM machine model and generates each $G \in \mathcal{D}(n, m, k)$ with positive probability.*

Proof. Let G be a well-ordered k-degenerate and let $v_1, \dots, v_n$ whose out-degree sequence is $d_1, \ldots, d_n$. Then Algorithm 3.6 generates G with probability at least

$$\frac{m!}{(n-1)^m} \cdot \prod_{i=1}^{n-1} \binom{n-i}{d_i}^{-1} .$$

To see this, consider a single pass of the first for-loop of the algorithm. Since C contains at most $n-1$ vertices, a fixed vertex $v_i \in C$ is chosen with probability at least $1/(n-1)$ in the current pass. Then the probability of choosing the vertices with correct multiplicities is at least $m!/(n-1)^m$. In the second phase of the algorithm, the probability of choosing the correct set of neighbors for each vertex v_i is at least $\binom{n-i}{d_i}^{-1}$. Hence, G is generated with positive probability and the algorithm is therefore complete.

Clearly, the loop in Line 5 can be implemented to run in time $\mathcal{O}(m)$ using an array for C and swapping deleted vertices to the back of the array. The loop in Line 10 can be implemented to run in time $\mathcal{O}(n+m)$ as follows: We maintain an array A of the indices $1, \ldots, n$ such that the array $A[i, \ldots, n]$ contains the number $i, \ldots, n$ in arbitrary order when dealing with vertex i. When choosing the j-th neighbor of v_i we pick a random number r from $i+j, \ldots, n$ and create an edge from v_i to $v_{A[r]}$. Then we swap $A[r]$ and $A[i+j]$. Clearly, the operations can be performed in constant time for each vertex and no duplicate edges are created. Hence, the algorithm can be implemented to run $\mathcal{O}(n+m)$ time. $\square$

Note that the algorithm does not generate well-ordered k-degenerate graphs according to a uniform distribution: In order to see this, we consider the sequence of successor-degrees induced by the well-ordering of the vertices of a k-degenerate graph. Any 2-degenerate graph with successor-degree sequence $1, 1, 0$ is generated with probability $3/8$ and the only 2-degenerate graph with successor-degree sequence $2, 0, 0$ is generated with probability $1/4$.

3.6 Generating Well-Ordered Strongly Degenerate Graphs

In this section we consider the problem of generating *well-ordered strongly k-degenerate graphs*. A graph is well-ordered strongly k-degenerate if it is well-ordered k-degenerate and if its minimum degree is k. Strongly k-degenerate graphs, that is, k-degenerate graphs with minimum degree k, are a natural generalization of k-regular graphs. As mentioned in the introduction, strongly k-degenerate graphs are also interesting regarding to their relation to the core hierarchy of a graph. We denote the class of well-ordered strongly k-degenerate graphs with n vertices by $\mathcal{S}(n,k)$ and the class of well-ordered strongly k-degenerate graphs with n vertices and m edges by $\mathcal{S}(n,m,k)$, respectively. The generators presented in the previous sections are based on the fact that, for any well-ordered k-degenerate graph G, the subgraphs $G_i := G[v_i, \ldots, v_n]$ are well-ordered k-degenerate. Although it remains true that the subgraphs G_i are well-ordered k-degenerate for any well-ordered *strongly* k-degenerate graph G, these subgraphs are not well-ordered *strongly* k-degenerate. In order to apply the decomposition used in the previous section, we would have to make sure that, whenever we create a vertex whose out-degree is $d < k$, we will be able to reserve $k-d$ edges for this vertex in future steps. Let the *deficiency* of a vertex v be defined as $\max\{k - \mathrm{d}(v), 0\}$ and let the deficiency of a well-ordered k-degenerate graph G be defined as

$$\Delta(G) := \sum_{v \in V} \max\{k - \mathrm{d}(v), 0\} .$$

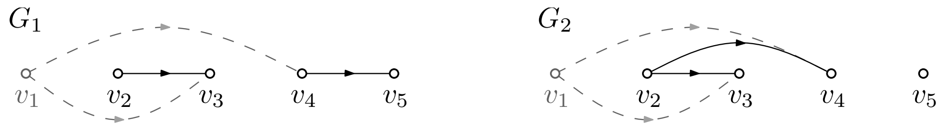


Figure 3.4: Two well-ordered 2-degenerate graphs G_1 and G_2 with deficiency 4 and vertex sets $v_2, \dots, v_5$. The graph G_1 can be augmented to a well-ordered 2-degenerate graph with deficiency 2 by a new vertex v_1 and 6 possible sets of edges incident to v_1, whereas, for G_2, there are only 3 possible sets of edges.

That is, the deficiency of a well-ordered k-degenerate graph is a measure for how close this graph is to being well-ordered strongly k-degenerate. Clearly, a well-ordered k-degenerate graph G is a well-ordered strongly k-degenerate graph if and only if $\Delta(G) = 0$. By applying the decomposition used in the previous section, we essentially decompose a well-ordered k-degenerate graph with deficiency δ into a vertex v with degree d_v and a well-ordered k-degenerate graph with deficiency δ'. In order to apply the same approach for well-ordered strongly k-degenerate graphs, we would have to compute, for a given values n, m, k and δ, a well-ordered k-degenerate graph with n vertices, m edges and deficiency δ.

Let $\mathcal{D}(n, m, k, \delta)$ denote the class of k-degenerate graphs with n vertices, m edges and deficiency δ and let $D(n, m, k, \delta)$ denote its cardinality. Let $0 \leq d \leq k$ and let $G' \in \mathcal{D}(n-1, m-d, k, \delta')$ for some deficiency δ'. By $A(G', n, m, k, \delta)$ we denote the number of possibilities to augment G' with d edges incident to a newly added vertex such that the resulting graph has n vertices, m edges and deficiency δ. Since the deficiency is bounded by nk we have

$$D(n, m, k, \delta) = \sum_{d=0}^{k} \left(\sum_{\delta'=0}^{nk} \left(\sum_{G' \in \mathcal{D}(n-1, m-d, \delta')} A(G', n, m, k, \delta) \right) \right) .$$

Unfortunately, $A(G', n, m, \delta)$ depends not only on d and δ', but also on the structure of the graph G'. More precisely, it depends on the deficiency sequence of the vertices of G'. Without knowing this sequence, we cannot tell whether adding a specific edge will reduce the number of vertices with positive deficiency. As an example consider the two 2-degenerate graphs on vertex set $v_2, \dots, v_5$ and deficiency four illustrated in Figure 3.4 that are to be augmented with a new vertex v_1 and two additional edges such that the resulting graph has deficiency two. For the leftmost graph, G_1, there are $\binom{4}{2} = 6$ possible augmentations, whereas for the rightmost graph, G_2, there are only $\binom{3}{2} = 3$ possible augmentations. This is due to the fact that the edge (v_1, v_2) can be used to decrease the deficiency of G_1, but not that of G_2, since in G_2, the degree of v_2 already equals two. Hence, it does not seem to be possible to simplify the above formula largely, which renders the approach used to generate k-degenerate graphs in the previous section useless for strongly k-degenerate graphs.

In order to overcome these difficulties we propose the following alternative approach for generating well-ordered strongly k-degenerate graphs: First we generate a random well-ordered k-degenerate graph with n vertices (and m edges). Then we transform this graph into a well-ordered strongly k-degenerate graph. We have seen how ordinary well-ordered k-degenerate graphs with n vertices (and m edges) can be generated in the previous sections. Next, we describe how an ordinary well-ordered k-degenerate graph can be transformed into a well-ordered strongly k-degenerate graph in $\mathcal{O}(kn^2)$ time.

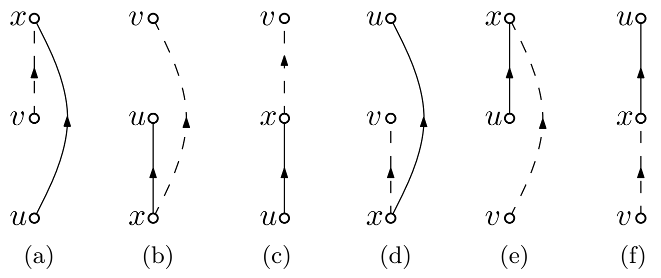


Figure 3.5: The possible orderings of u, v, x in Lemma 3.8: dashed edges are replaced by solid edges.

Lemma 3.8. *Let $G = (V, E)$ be a well-ordered k-degenerate graph with n vertices, $m \geq \frac{nk}{2}$ edges and deficiency $\Delta(G) > 0$. Then there is an edge $\{v, x\} \in E$ and a vertex u such that $G' := (G - \{v, x\}) + \{u, x\}$ is well-ordered k-degenerate and $\Delta(G') < \Delta(G)$.*

Proof. Since $\Delta(G) > 0$ there must be at least one vertex u such that $\mathrm{d}(u) < k$. On the other hand, there must be a vertex v such that $\mathrm{d}(v) > k$. Otherwise

$$m = \frac{1}{2}\sum_{v\in V}\mathrm{d}(v) = \frac{1}{2}\left(\sum_{v\in V\setminus\{u\}}\mathrm{d}(v) + \mathrm{d}(u)\right) < \frac{(n-1)k + k}{2} = \frac{nk}{2}$$

in contradiction to the assumption that $m \geq \frac{nk}{2}$.

Let u be an arbitrary vertex such that $d(u) < k$ and let v be the largest vertex such that $\mathrm{d}(v) > k$. Since $\mathrm{d}(v) > k$ and $\mathrm{d}(u) < k$ there is a vertex $x \in N(v) \setminus N(u)$. Hence $G' := (G - \{v, x\}) + \{u, x\}$ is simple and

$$\Delta(G') = \sum_{v\in V'\setminus\{u,v\}} \max\{k - \mathrm{d}_G(v), 0\} + \max\{k - (\mathrm{d}_G(u) + 1), 0\} = \Delta(G) - 1 \ .$$

It remains to show that G' is k-degenerate. Recall that a well-ordered graph is k-degenerate if and only if all vertices v satisfy $\mathrm{d}^+(v) \leq k$. Note that this property can only be violated by edge-insertions (but not by edge-deletions), that is, if the successor-degree sequence is increased for some vertex. Further $\mathrm{d}(u) < k$ implies $\mathrm{d}^+(u) < k$. There are six possible orderings of the vertices u, v, x. We will consider three cases:

Case (i): $x < u, v$. The successor-degree sequence is not changed by the operation as illustrated in Figures 3.5b and 3.5d.

Case (ii): $u < x$. The successor-degree is increased only for u. Since $\mathrm{d}^+_G(u) < k$ we obtain $\mathrm{d}^+_{G'}(u) \leq k$ as illustrated in Figures 3.5a, 3.5c, and 3.5e.

Case (iii): $v < x < u$. If $\mathrm{d}^+_G(x) < k$ then $\mathrm{d}^+_{G'}(x) \leq k$ as illustrated in Figure 3.5f. Otherwise $\mathrm{d}^+(x) = k$. Since we have assumed that v is the largest vertex with degree greater than k we have $\mathrm{d}(x) \leq k$, that is, $\mathrm{d}(x) = k$. However, this implies $\mathrm{d}^-(x) = 0$ in contradiction to the assumption that v is a neighbor of x and $v < x$.

This concludes the proof. □

In order to obtain a canonical transformation we may assume, without loss of generality, that x and u are the smallest (respectively largest) vertices with the desired properties. Using this Lemma, we obtain efficient and complete generators for well-ordered strongly k-degenerate graphs in $\mathcal{S}(n,k)$ and $\mathcal{S}(n,m,k)$, respectively. The pseudo-code is listed in Algorithm 3.7. The Algorithm generates a random well-ordered k-degenerate graph and repeatedly applies Lemma 3.8 in order to transform the well-ordered k-degenerate graph into a well-ordered strongly k-degenerate graph.

Theorem 3.8. *Algorithm 3.7 generates each well-ordered strongly k-degenerate graph with n vertices (and $m \geq \lceil \frac{nk}{2} \rceil$ edges) with positive probability in time $\mathcal{O}(n^2k + n\log n)$ and $\mathcal{O}(nmk + n\log k)$, respectively.*

Proof. Since $\mathcal{S}(n,k) \subset \mathcal{D}(n,k)$ and $\mathcal{S}(n,m,k) \subset \mathcal{D}(n,m,k)$, respectively, Algorithm 3.7 generates each well-ordered strongly k-degenerate graph with n vertices (and m edges) with positive probability.

The first step of the algorithm can be performed in time $\mathcal{O}(n\log k + m)$ for graphs with given number of vertices and in time $\mathcal{O}(n\log k + m + nk)$ for graphs with given number of vertices and edges. The algorithm then terminates after exactly $\Delta(G)$ applications of Lemma 3.8, where G denotes the graph generated in Line 1. For a well-ordered k-degenerate graph G with n vertices $\Delta(G) \leq nk$. An application of Lemma 3.8 can be implemented to run in $\mathcal{O}(n)$ time. Therefore, the algorithm can be implemented to run in $\mathcal{O}(n^2k + n\log k)$ and $\mathcal{O}(nmk + n\log k)$ time, respectively. $\square$

Note that the transformation of k-degenerate graphs using Lemma 3.8 is a surjective function (projection)

$$\pi : \mathcal{D}(n,m,k) \to \mathcal{S}(n,m,k)$$

of the k-degenerate graphs onto the strongly k-degenerate graphs. If the uniform generator for ordinary k-degenerate graphs is used in the first step of the algorithm then the probability of obtaining a fixed strongly k-degenerate graph G is given by

$$\mathbb{P}(G) = \frac{|\pi^{-1}(G)|}{D(n,m,k)}\ .$$

Algorithm 3.7: STRONGLY-DEGENERATE($n[, m]$)

```
Input: $n \in \mathbb{N}$, $0 \leq k \leq n[$, $\frac{nk}{2} \leq m \leq M(n,k)]$
Output: well-ordered strongly $k$-degenerate graph with $n$ vertices [and $m$ edges]
1 $G \leftarrow$ GENERATE-DEGENERATE($n[, m], k$)
2 while $\Delta(G) > 0$ do
3     $u \leftarrow$ smallest vertex with $\mathrm{d}(u) < k$
4     $v \leftarrow$ largest vertex with $\mathrm{d}(v) > k$
5     $x \leftarrow$ smallest vertex in $N(v) \setminus N(u)$
6     $G \leftarrow (G - \{v,x\}) + \{u,x\}$
7 return $G$
```

Clearly, $\pi^{-1}(G)$ tends to be smaller if the deficiency of G is small. That is, by rejecting graphs with high deficiency we obtain a generator whose distribution is closer to the uniform distribution. At the cost of running time we can get arbitrary close to the uniform distribution.

Although Algorithm 3.7 can be used to generate strongly k-degenerate graphs with n vertices, its running time is rather slow. Next, we therefore describe a faster complete non-uniform algorithm for generating strongly k-degenerate graphs with n vertices at random improving on the previous algorithm. Unfortunately, however, this algorithm cannot be used to generate strongly k-degenerate graphs with given number of vertices and edges, respectively. For these graphs it is not clear how to improve on Algorithm 3.7.

The algorithm we propose for generating strongly k-degenerate graphs with given number is summarized in Algorithm 3.8 and proceeds as follows. We start with an empty graph adding vertices one at a time. In the i-th step we insert vertex v_{n-i+1} and a set of edges incident to v_{n-i+1} such that the resulting graph can be augmented to a strongly k-degenerate graph with n vertices by inserting $i-1$ additional vertices with degree at most k. We call a well-ordered k-degenerate graph with vertices $v_{i+1}, \ldots, v_n$ i-conditioned, if it can be augmented to a well-ordered strongly k-degenerate graph with n vertices by adding i vertices $v_1, \ldots, v_i$ and, for each vertex v_s with $1 \leq s \leq i$, a set of at most k edges (v_s, v_t) with $t > s$. The following lemma characterizes i-conditioned graphs based on the vertex degrees and the deficiency of the graph.

Lemma 3.9. *Let $n \in \mathbb{N}$ and let $k \in \mathbb{N}$ such that $n \geq k+1$. A well-ordered k-degenerate graph G_{i+1} with vertices $v_{i+1}, \ldots, v_n$ is i-conditioned if and only if*

(i) $\mathrm{d}(v_j) \geq k - i$ *for all $i+1 \leq j \leq n$ and*

(ii) $\Delta(G_{i+1}) \leq ik$.

Proof. First, suppose that G_{i+1} is i-conditioned, that is, it can be augmented to a well-ordered strongly k-degenerate graph G by adding a set E_i of edges incident to a set of newly added vertices $v_1, \ldots, v_i$ such that the out-degree of each newly added vertex is at most k. Since each newly added vertex v_s with $1 \leq s \leq i$ has at most one edge to each vertex v_t with $i+1 \leq t \leq n$ and since, after the augmentation, the degree of each vertex is at least k, we have (i) $\mathrm{d}(v_t) \geq k - i$ for all $i+1 \leq t \leq n$. Further, since $\Delta(G) = 0$ and since each vertex v_s with $1 \leq s \leq i$ introduces at most k new edges, each of which can reduce the deficiency by at most one unit, we have (ii) $\Delta(G_{i+1}) \leq ik$.

Second, suppose that (i) and (ii) hold. We show that G_{i+1} can be augmented to a well-ordered strongly k-degenerate graph with n vertices by adding a set of vertices $v_1, \ldots, v_i$ and a set of edges E_i incident to these vertices. We prove this by induction on i. For $i = 0$, the graph G_1 has deficiency 0 by (ii), which implies that G_1 is strongly k-degenerate with n vertices. Thus, we are done if we can iteratively augment the graphs satisfying (i) and (ii). First we show that, for $i = n$, the graph G_n satisfies (i) and (ii). The graph G_n consists of a single vertex whose degree is zero. Clearly, the degree of v_n is at least $k - n$ which is at least $k - k = 0$, that is, G_n satisfies (i) and the deficiency of G_n is clearly equal to k, which is less than k^2 for all values of k, that is G_n also satisfies (ii). Thus, the induction hypothesis holds for the base case.

Suppose we are given a graph G_{i+1} satisfying (i) and (ii). We show how to augment G_{i+1} by a single vertex and some edge incident to this vertex such that the resulting graph G_i also satisfies (i) and (ii). Let $U_1 \subseteq V_{i+1}$ denote the set of vertices with degree exactly $k - i$, that is,

$$U_1 := \{v \in V_{i+1} \mid \mathrm{d}(v) = k - i\}$$

and let $U_2 \subseteq V_{i+1}$ denote the set of vertices with degree greater than $k - i$, but less than k that is,

$$U_2 := \{v \in V_{i+1} \mid k - i < \mathrm{d}(v) < k\} \ .$$

Then the set $U := U_1 \uplus U_2$ contains exactly the vertices in V_{i+1} with positive deficiency that must be augmented by additional edges. All other vertices need not be augmented by further edges. Due to the induction hypothesis and due to (i) we have $\mathrm{d}(v) \geq k - i$ for all $v \in V_{i+1} := \{v_{i+1}, \ldots, v_n\}$. Together with (ii) this implies that there are at most k vertices in U_1, since $|U_1| > k$ would imply $\Delta(G_{i+1}) > ik$, a contradiction to (ii). We distinguish two cases.

First, suppose that $1 \leq i \leq k$. We show that we can always choose a set of vertices W such that the graph resulting from introducing a new vertex v_i and connecting v_i to all vertices in W will meet the conditions (i) and (ii). Let $W \subseteq V_{i+1}$ be such that the following holds

(a) The set W contains enough vertices from U to decrease the deficiency such that (ii) will be met, that is $|U \cap W| \geq \Delta(G_{i+1}) - (i - 1)k$.

(b) The set W contains all vertices in U_1 that must necessarily be augmented such that (i) will be met, that is, $U_1 \subseteq W$.

(c) The set W contains enough vertices to raise the degree of v_i to $k - i + 1$ such that (i) is met for v_i, that is, $|W| \geq k - i + 1$, but not more than $\min\{n - i, k\}$ since the resulting graph must be well-ordered k-degenerate, that is $|W| \leq \min\{n - i, k\}$.

We show that a set W with the desired properties always exists. Note that

$$\Delta(G_{i+1}) \leq |U_1|i + |U_2|(i - 1) < |U|i \tag{3.15}$$

by the definition of the deficiency and the definition of U_1 and U_2, respectively. Suppose for contradiction that $|U| < \Delta(G_{i+1}) - (i - 1)k$. Then, by Equation (3.15) we have

$$\begin{aligned}\Delta(G_{i+1}) &< |U|i \\ &< (\Delta(G_{i+1}) - (i - 1)k)i \ .\end{aligned}$$

For $i > 1$, this is equivalent to

$$(i - 1)\Delta(G_{i+1}) > i(i - 1)k,$$

that is, $\Delta(G_{i+1}) > ik$ in contradiction to the assumption that (ii) holds. Hence, for $i > 1$, we have $U \geq \Delta(G_{i+1}) - (i - 1)k$, which implies that we can always satisfy (a). For $i = 1$ we have $\mathrm{d}(v_j) \geq k - 1$ for all $2 \leq j \leq n$ by (i), that is, $U_2 = \emptyset$ and, thus, $U = U_1$. Further, (ii) implies $|U| = |U_1| = \Delta(G_{i+1})$. Hence, in this case, we can also satisfy (a) since we argued that $U_1 \leq k$. Clearly, $|U_1| \leq n - i$ since $U_1 \subseteq V_{i+1}$ that is, including U_1 in W does not violate the upper bound on $|W|$ required by (c).

Additionally, since $U_1 \leq \min\{n - i, k\}$ we can also satisfy (b) without violating the upper bound on $|W|$ required by (c). Finally, if $i \leq k$, then $|V_{i+1}| \geq k - i + 1$. Otherwise we would have $n < i + k - i + 1 = k + 1$, a contradiction to the assumption that $n \geq k + 1$. That is, we can also satisfy the lower bound required by (c) and we can therefore always find a set W with the properties stated above. In the following, let W be any set with the desired properties.

Let G_i be the graph resulting from adding the vertex v_i and the edges

$$E_i := \{(v_i, w) \mid w \in W\}$$

to G_{i+1}. We claim that (i') $\mathrm{d}(v) \geq k-i+1$ for all $v \in V_i := V_{i+1} \cup \{v_i\}$ and (ii') $\Delta(G_i) \leq (i-1)k$. To see (i'), note that $\mathrm{d}(v_i) = |W| \geq k-i+1$ by (c) and that $U_1 \subseteq W$, that is we augmented the degree of all vertices with minimum degree $k-i$ in G_{i+1} such that the minimum degree is $k-i+1$ in G_i due to (b). Finally, (ii') follows from the fact (a) that $|U \cap W| \geq \Delta(G_{i+1}) - (i-1)k$. For each vertex in $|U \cap W|$ the deficiency is decreased such that the resulting deficiency of G_i is at most $(i - 1)k$. Further, since $|W| \leq \min\{n - i, k\}$ the resulting graph is well-ordered k-degenerate.

Second, suppose that $i > k$. Let $W \subseteq V_{i+1}$ such that $|U \cap W| \geq \Delta(G_{i+1}) - (i-1)k$, that is, W satisfies (a). Note that $U_1 = \emptyset$ for $i > k$ since this implies $k - i < 0$, hence, W trivially satisfies (b). Similarly, $k - i + 1 \leq 0$, that is, W also trivially satisfies (c). We have proven the existence of such a set W satisfying (a) in the previous case, without using $i \leq k$, which we used only for (b) and (c). Again, let G_i be the graph resulting from adding the vertex v_i and the edges

$$E_i := \{(v_i, w) \mid w \in W\}$$

to G_{i+1} and let (i') and (ii') be as in the previous case. Since $k - i + 1 \geq 0$ for $i > k$, (i') is trivially true for G_i and (ii') follows analogous to the previous case. □

The previous lemma is constructive in that it provides a characterization of the sets of edges that can be used to augment an i-conditioned graph to an $(i - 1)$-conditioned graph using a single new vertex and some edges incident to this vertex. We use this idea to generate strongly k-degenerate graphs with $n \geq k + 1$ vertices as follows.

We start with a graph G_n consisting of a single vertex v_n, which is clearly $(n - 1)$-conditioned, since $\mathrm{d}(v_n) = 0 \geq k - n - 1$ due to $n \geq k + 1$ and since $\Delta(G_n) = k$. Then we iteratively add a single vertex and a set of edges W with the properties used in the proof of Lemma 3.9. Suppose that we are given an $(i + 1)$-conditioned graph as in the proof of the lemma and we wish to construct and i-conditioned graph G_i. We choose a set of vertices W as follows. First, we select all vertices U_1 with degree $k - i$ in G_{i+1}. Then we randomly choose a set Z of $\Delta(G_{i+1}) - (i - 1)k - |U_1|$ vertices from the set U_2 of vertices with positive deficiency that are not contained in U_1. Finally, we choose a random number r between $\max\{0, k - i + 1\}$ and $\min\{n - i, k - (|U_1| + |Z|)\}$ and randomly choose a set X of r additional vertices from $V_{i+1} \setminus W$. Then we add the vertex v_i and the edges

$$E_i := \{(v_i, w) \mid w \in U_1 \cup Z \cup X\}$$

Algorithm 3.8: DEGENERATE(n, k)

```
Input: $k \in \mathbb{N}$, $n \in \mathbb{N}$ such that $n \geq k + 1$
Output: random strongly $k$-degenerate graph with $n$ vertices
1  $V \leftarrow \{v_n\}$
2  $E \leftarrow \emptyset$
3  $U_2 \leftarrow \emptyset$
4  if $k > 0 \wedge n > k + 1$ then
5      $U_2 \leftarrow U_2 \cup \{v_n\}$
6  for $i = n - 1$ to $1$ do
7      $V \leftarrow V \cup \{v_i\}$
8      $U_1 \leftarrow \{w \in V \mid \mathrm{d}(w) = k - i\}$
9      $E \leftarrow E \cup \{(v_i, w) \mid w \in U_1\}$
10     $Z \leftarrow$ randomly choose $\Delta(G_{i+1}) - (i-1)k - |U_1|$ edges from $U_2$
11     $r \leftarrow$ random integer between $\max\{0, k - i + 1\}$ and $\min\{n - i, k - (|U_1| + |Z|)\}$
12     $X \leftarrow$ randomly choose $r$ edges from $V \setminus (U_1 \cup Z)$
13     for $w \in Z \cup X$ do
14         $E \leftarrow E \cup \{(v_i, w)\}$
15         if $\mathrm{d}(w) = k$ then
16             remove $w$ from $U_2$
17     if $\mathrm{d}(v_i) < k \wedge \mathrm{d}(v_i) > k - i + 1$ then
18         $U_2 \leftarrow U_2 \cup \{v_n\}$
19 return $G = (V, E)$
```

to G_{i+1} to obtain G_i. Since $W := U_1 \uplus Z \uplus X$ satisfies the conditions (a)–(c) required from W in the proof of Lemma 3.9, G_i is i-conditioned as desired. The algorithm summarized in Algorithm 3.8.

Theorem 3.9. *Algorithm 3.8 generates each well-ordered strongly k-degenerate graph with n vertices with positive probability in time $\mathcal{O}(n + m)$ in the RAM machine model.*

Proof. Note that the sets used in the algorithm are defined analogous to the sets used in the proof of Lemma 3.9 with $W := U_1 \uplus Z \uplus X$.

First, we show that the algorithm only computes well-ordered strongly k-degenerate graphs. To see why the generated graph is well-ordered k-degenerate it suffices to note that each vertex has out-degree at most k. The out-degree of v_i is exactly the size of $U_1 \uplus Z \uplus X = W$. Observe that, due to Line 11, we have $|X| \leq k - |U_1 \uplus Z|$. Hence, it suffices to show that $|U_1 \uplus Z| \leq k$. We already mentioned that $U_1 \leq k$. Further $\Delta_{i+1} - (i-1)k \leq k$. Hence $U_1 \uplus Z$ indeed contains at most k elements and, therefore, the degree of v_i is at most k. Since the selected set of vertices is contained in V_{i+1} it is trivially bounded by $n - i$.

In order to show that the graph returned by the algorithm is *strongly* k-degenerate it suffices to show that the graph $G = (V, E)$ maintained by the algorithm is $(i-1)$-conditioned after the j-th iteration where $i = (n - j)$. However, this follows analogous to the proof of Lemma 3.9 by induction over i. As argued before, G_n is $(n-1)$-conditioned and since we

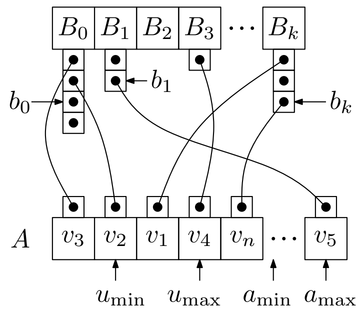


Figure 3.6: Data structure used for the implementation of Algorithm 3.8. The array I is omitted for reasons of clarity.

inserted a set of edges satisfying conditions (a)–(c) of the proof of Lemma 3.9, the graph constructed in iteration i is $(i-1)$-conditioned.

Second, we show that each well-ordered strongly k-degenerate graph is generated with positive probability. Let G be a well-ordered strongly k-degenerate graph with n vertices. Then for each i the graph $G_i := G[v_i, \dots, v_n]$ is $(i-1)$-conditioned. Removing the edges incident to $v_1, \dots v_{i-1}$ results a total deficiency of G_i of at most $(i-1)k$. Further, the deficiency of each vertex is at most $i-1$, hence, its degree is at least $k-i+1$. By induction we show that G_i corresponds to a graph that is generated with positive probability in the $(n-i+1)$-th iteration. Clearly, in the first iteration we have $i = n$ and G_n corresponds to the graph that is deterministically constructed in the first step of the algorithm. In the induction step, suppose that G_{i+1} corresponds to some graph that is generated in the $(n-i)$-th iteration of the algorithm with positive probability and consider G_i. Let $E_i := E(G_i) \setminus E(G_{i+1})$. By Lemma 3.9, E_i must contain all edges in U_1 as well as at least $\Delta(G_{i+1}) - (i-1)k - |U_1|$ edges from U_2. Otherwise, either the total deficiency of G_i would be larger than ik or there would be a vertex with degree less than $k-i$ in G_i. The remaining edges are arbitrary. Thus, the set of edges E_i is chosen with positive probability in iteration i, which proves the claim.

The algorithm can be implemented to run in $\mathcal{O}(n+m)$ time as follows. We store the vertices in an array A with random access in order to be able to sample vertices by picking a random integer. Instead of adding the vertices one at a time, we pre-fill the array with the vertices $v_1, \dots, v_n$. The set U_2 used in the algorithm is maintained as a sub-array of A consisting of a prefix of A. Let $u_{\min}$ and $u_{\max}$ denote the minimum and maximum index of an element of U_2 in A and let $a_{\min}$ and $a_{\max}$ denote the minimum and maximum index of an element of $V \setminus U_2$, respectively. Since the elements in A will not be maintained in their correct order throughout the algorithm, we additionally maintain an array I such that the i-th element of I contains the index of v_i in A. For each degree i we further maintain an array B_i of pointers to the positions of vertices with degree i in A as well as an index b_i pointing to the last valid element in B_i. An element in B_i is called *valid* if we can insert an edge from v_i to the corresponding vertex in the current iteration without creating multiple edges between these two vertices. For each vertex v in A we also store a pointer to its position in the array $B_{\mathrm{d}(v)}$. The data structure, excluding I, is illustrated in Figure 3.6.

When adding a vertex v_i in Line 7, we locate the index j of v_i in A by inspecting the i-th entry of I, then we decrease $a_{\min}$ by one unit and swap the elements $a_{\min}$ and j in A, updating the pointers in I and the corresponding B-arrays.

Note that we need not maintain U_1 explicitly, since U_1 is empty for all iterations with $i \geq k$ and since it corresponds exactly to the vertices in B_{k-i} in iteration $i < k$. When creating the edges according to Line 9 we therefore iterate over the elements in B_{k-i} starting with the last element. In each step, we remove the last element from B_{k-i} and add it to the end of B_{k-i+1} without increasing b_{k-i+1}, since we have already created an edge, such that the newly added vertices are not valid in the current iteration anymore. Before proceeding to the next iteration of Algorithm 3.8 we set b_{k-i+1} to the size of B_{k-i+1}. Additionally, we maintain the pointers in A accordingly, that is, whenever an element is moved, its pointer is updated in A.

Whenever a vertex v_i is added to U_2, we proceed analogous to adding a new vertex to V. However, instead of decreasing $u_{\min}$ we increase $u_{\max}$ and swap the elements $u_{\max}$ and j, where j is the index of v_i in A. If v_i was located between $a_{\min}$ and $a_{\max}$, we additionally swap the elements $a_{\min}$ and j and increase $a_{\min}$. Hence, V is represented by the vertices in A between the indices $u_{\min}$ and $u_{\max}$ and $a_{\min}$ and $a_{\max}$ of A, respectively. During the swapping the pointers to the elements in A are maintained accordingly.

When picking an element w according to Line 10 of the algorithm, we pick a random index j between $u_{\min}$ and $u_{\max}$. Then we swap the elements j and $u_{\min}$ and increase $u_{\min}$. At the end of the iteration, we re-set $u_{\min} = 1$. Elements at the beginning of A belong to U_2 and have already been chosen as the endpoint of and edge and will not be chosen again. We proceed similarly, when choosing a vertex w according to Line 12. However, in this case we pick a random index j between $u_{\min}$ and $u_{\max}$ or between $a_{\min}$ and $a_{\max}$, respectively. If j is between $a_{\min}$ and $a_{\max}$, we swap the j-th element with $a_{\max}$ and decrease $a_{\max}$ such that no element is chosen more than once. At the end of the iteration, we re-set $a_{\max} = n$. Clearly, all these operations are supported in $\mathcal{O}(1)$ time by the proposed data structure. The deficiency is updated on the fly after each operation. Hence, each step of the algorithm can be performed in constant time, which yields a total complexity of $\mathcal{O}(n + m)$. □

3.7 Enumerating Well-Ordered Degenerate Graphs

In this section we show how to enumerate well-ordered strongly k-degenerate graphs with n vertices and with n vertices and m edges, respectively. The presented algorithms can be modified in a straight-forward way in order to enumerate ordinary well-ordered k-degenerate graphs. Throughout this section, we assume the RAM machine model. When designing an enumeration algorithm for a class of graphs, it is clearly desirable that all of the graphs is enumerated exactly once. That is, we need to show that the algorithm is complete and no graph is enumerated more than once. In order to facilitate this, it is common to represent the graphs in a canonical way. For well-ordered k-degenerate graphs, we can do this by representing a graph $G = (V, E)$ with $V = \{v_1, \dots, v_n\}$ as follows. For each vertex v_i we let $N^+(v_i)$ denote the *ordered* set of successors in the set $\{v_{i+1}, \dots, v_n\}$. We assume that the vertices in $N^+(v_i)$ are ordered according to the ordering induced by $v_1, \dots, v_n$. Then we obtain a canonical representation by encoding G as the vector

$$\gamma(G) := \Big(N^+(v_1), \dots, N^+(v_n)\Big) \ .$$

Clearly, two well-ordered k-degenerate graphs G and G' are equal if and only if $\gamma(G) = \gamma(G')$. Further, note that this representation induces a canonical ordering on the graphs by considering the lexicographical ordering on the vectors defined by γ. We say that G is *lexicographically* smaller (respectively, larger) than G' if $\gamma(G)$ is lexicographically smaller (larger) than $\gamma(G')$.

Let G be a well-ordered strongly k-degenerate graph. Then G has at least $n \geq k+1$ vertices. Otherwise the maximum degree of a vertex is bounded by $k-1$. Further, since each vertex has degree at least k, it has at least

$$m = \frac{1}{2}\sum_{i=1}^{n} \mathrm{d}(v_i) \geq \frac{nk}{2}$$

edges. A strongly k-degenerate graph with n vertices and $m(n,k) := \left\lceil \frac{nk}{2} \right\rceil$ edges is called *minimal.* If n or k is even, these graphs are exactly the k-regular graphs with n vertices. If both n and k are odd each minimal strongly k-degenerate graph with n vertices has exactly one vertex with degree $k+1$. To see that minimal well-ordered k-degenerate graphs exist, we can apply Lemma 3.8 to any well-ordered k-degenerate graph with $\lceil \frac{nk}{2} \rceil$ edges. Since the lemma states that we can reduce the deficiency as long as it is positive, this implies that we can transform any well-ordered k-degenerate graph with $\lceil \frac{nk}{2} \rceil$ edges and positive deficiency into a graph with the same number of edges and deficiency 0, that is, into a strongly k-degenerate graph. Thus, it suffices to show that we can always construct a well-ordered k-degenerate graph with n vertices and $\frac{nk}{2}$ edges. In order to construct a well-ordered k-degenerate graph with $\lceil \frac{nk}{2} \rceil$ edges we proceed as follows. We start with an empty graph consisting of vertices $v_1, \ldots, v_n$ and consider the vertices $v_1, \ldots, v_n$ in this order. Then we iteratively add single edges as long as the graph has less than $\lceil \frac{nk}{2} \rceil$ edges. When considering vertex v_i we either add another edge incident to v_i if the degree of v_i is at most $\min\{n-i, k\}$ or we proceed to the next vertex. Since each vertex v_i has at least $\min\{n-i,k\}$ possible neighbors and since the degree of each vertex v_i is bounded by $\min\{n-i,k\}$ we thus construct a well-ordered k-degenerate graph without multiple edges between the vertices. Further, since $n \geq k+1$ we have

$$\sum_{i=1}^{n} \min\{n-i,k\} = M(n,k) = nk - \frac{k(k+1)}{2} \geq nk - \frac{nk}{2} = \frac{nk}{2}$$

according to Equation 3.14. That is, the algorithm produces a well-ordered k-degenerate graph with at least $\frac{nk}{2}$ edges.

3.7.1 Enumerating Well-Ordered Degenerate Graphs with n Vertices

At first, we consider the problem of enumerating k-degenerate and strongly k-degenerate graphs with given number of vertices and arbitrary number of edges. The enumeration algorithm outlines as follows: At first we generate all *maximal* well-ordered k-degenerate graphs with n vertices, that is, all k-degenerate graphs with n vertices and $M(n,k)$ edges. For each maximal well-ordered k-degenerate graph G we identify and enumerate those well-ordered strongly k-degenerate subgraphs whose lexicographically smallest well-ordered k-degenerate supergraph is exactly G. Thus, we generate each graph exactly once.

At first we describe how to generate all maximal well-ordered k-degenerate graphs. We use the following observation.

Lemma 3.10. *Let G be a maximal well-ordered k-degenerate graph with $n \geq k+1$ vertices and let $G' := G[v_{n-k}, \dots, v_n]$. Then G' is a clique on $k+1$ vertices. Thus, any maximal well-ordered k-degenerate graph is strongly k-degenerate.*

Proof. If G is a maximal well-ordered k-degenerate graph, then $\mathrm{d}^+(v_i) = \min\{n-i, k\}$ for all $1 \leq i \leq n$. Suppose for contradiction that the $\mathrm{d}^+(v_j) < \min\{n-j, k\}$ for some $1 \leq j \leq n$. Then the number of edges of G is

$$m = \sum_{i=1}^{n} \mathrm{d}^+(v_i) < \sum_{i=1}^{n} \min\{n-i, k\} = M(n,k)$$

by Equation (3.14). Hence, G is not maximal in contradiction to the assumption, which implies $\mathrm{d}^+(v_i) = \min\{n-i, k\}$ for all $1 \leq i \leq k$. That is, $\mathrm{d}(v_i) \geq \mathrm{d}^+(v_i) = k$ for all $1 \leq i \leq n-k-1$. Further, consider a fixed vertex v_i such that $n-k \leq i \leq n$. Then $\mathrm{d}^+(v_i) = n-i$ and v_i has an incoming edge from each vertex v_j such that $n-k \leq j \leq i-1$. To see this, note that each vertex v_j with $n-k \leq j \leq i-1$ has out-degree $n-j$ and exactly $n-j$ vertices $v_{j+1}, \dots, v_n$ as potential neighbors. Therefore, v_j must have an edge to each of the vertices $v_{j+1}, \dots, v_n$, including v_i. Hence, the in-degree of v_i is at least $i-(n-k)$. It follows that

$$\mathrm{d}(v_i) = \mathrm{d}^+(v_i) + \mathrm{d}^-(v_i) = (n-i) + (i-(n-k)) = k\ .$$

Thus, the degree of all vertices is at least k, which proves that G is well-ordered strongly k-degenerate. Further, since G' is a simple directed graph such that $\mathrm{d}(v_i) = k$ and $\mathrm{d}^+(v_i) = n-i$ for all $n-k \leq i \leq n$, G' is a directed clique with $k+1$ vertices. □

For an integer i, let C_i denote the set $\{v_{i+1}, \dots, v_n\}$. Using the previous lemma we obtain the following characterization of the maximal well-ordered (strongly) k-degenerate graphs.

Lemma 3.11. *Let $\mathcal{M}(n)$ be the set of all well-ordered maximal k-degenerate graphs with n vertices. Then there is a bijection*

$$\Psi_{n,k} : \mathcal{M}(n) \to \bigtimes_{i=1}^{n-k-1} \binom{C_i}{k}\ . \tag{3.16}$$

That is, there is a total of

$$\prod_{i=1}^{n-k-1} \binom{n-i}{k} \tag{3.17}$$

maximal (strongly) k-degenerate graphs.

Proof. Let G be a well-ordered k-degenerate graph and let $N^+(v_i)$ be defined as the set of neighbors of v_i with index greater than i. We define

$$\Psi_{n,k}(G) := (N^+(v_i))_{i=1}^{n-k-1}\ .$$

Clearly, $\Psi_{n,k}$ is injective since the last $k+1$ vertices of a maximal well-ordered k-degenerate graphs form a clique and need not be explicitly encoded by Lemma 3.10. To show that $\Psi_{n,k}$ is surjective let

$$(X_i)_{i=1}^{n-k-1} \in \bigtimes_{i=1}^{n-k-1} \binom{C_i}{k} .$$

Then $\Psi_{n,k}^{-1}(x)$ corresponds to a graph $G = (V, E)$ such that $V = \{v_1, \dots, v_n\}$ and

$$E = \{\{v_i, x\} \mid 1 \le i \le n-k-1,\ x \in X_i\} \cup \{\{v_i, v_j\} \mid n-k \le i < j \le n\} .$$

Since every vertex v_i satisfies $\mathrm{d}^+(v_i) = \mathrm{d}^+_{\max}(v_i) := \min\{n-i, k\}$ the graph G is a maximal well-ordered k-degenerate graph. Equation (3.17) follows directly from Equation (3.16). □

This characterization can be used to efficiently enumerate all maximal well-ordered k-degenerate graphs with n vertices.

Theorem 3.10. *There is an algorithm that enumerates all maximal well-ordered k-degenerate graphs with n vertices in amortized time $\mathcal{O}(n+m)$ per enumerated graph.*

Proof. If $n = k+1$, we construct a directed clique with $k+1$ vertices and we are done. For the remainder of the proof we therefore assume that $n > k+1$. The representation given in Equation (3.16) can also be encoded by a sequence of strings as follows. Each set $A \in \binom{C_i}{k}$ can be represented by a string $s(A)$ consisting of $|C_i| - k$ zeros and k ones such that the j-th character in $s(A)$ is equal to one if and only if $v_{i+j} \in A$. Then each graph can be represented by the concatenation of these strings. Ruskey and Williams [RW09] show how to enumerate all possible binary strings consisting of $|C_i| - k$ zeros and k ones in constant amortized time. We can use this algorithm as a sub-routine in an algorithm similar to a counter with $n-k-1$ digits such that the i-th digit assumes the possible strings $s(A)$ for $A \in C_i$. An augmentation corresponds to computing the next string according to the algorithm of Ruskey and Williams for some digit and possibly re-setting some of the digits to the string that forms the basis of the algorithm by Ruskey and Williams, just like for an ordinary counter. Note that every re-set of a digit is preceded by an augmentation of the digit and that every augmentation corresponds to a newly enumerated maximal well-ordered k-degenerate graph. Hence the re-set operations can be amortized over the enumerated graphs. Thus, we can enumerate and print every maximal well-ordered k-degenerate graph in $\mathcal{O}(n+m)$ amortized time per printed graph. □

For a well-ordered k-degenerate graph G we denote the lexicographically smallest maximal well-ordered k-degenerate supergraph by $S(G)$. A well-ordered k-degenerate graph G with n vertices is called a *proper subgraph* of a maximal well-ordered k-degenerate graph S with n vertices if and only if $S = S(G)$, that is, if S is the lexicographically smallest maximal k-degenerate supergraph of G. We observe that the lexicographically smallest maximal well-ordered k-degenerate supergraph of G can be obtained by including the $\mathrm{d}^+_{\max}(v_i) - \mathrm{d}^+(v_i)$ lexicographically smallest edges for every vertex v_i that are not yet incident to v_i. Let $X(G)$ denote this set of edges. Given a graph G, the *foundation* $F(v_i)$ of a vertex v_i is the largest set of edges $\{v_i, v_{i+1}\}, \dots, \{v_i, v_{i+\ell}\}$ such that $F(v_i) \in E$. That is, if v does not contain e_1 then $F(v) = \emptyset$. The foundation of a graph G is defined as $F(G) = \bigcup_{v \in V} F(v)$. We state the following:

Algorithm 3.9: ENUMERATE-FOR-GRAPH(G, X, S)

```
Input: well-ordered (strongly) $k$-degenerate graph $G$, forced edges $X$, supergraph $S$
Output: all well-ordered (strongly) $k$-degenerate proper subgraphs of $S$ containing no
        edges in $X$
1 if $(F(S) \cap E(G)) \setminus X \neq \emptyset$ then
2     $e \leftarrow$ choose an edge in $(F(S) \cap E(G)) \setminus X$
3     if $G - e$ is well-ordered (strongly) $k$-degenerate then
4         print $G - e$
5         ENUMERATE-FOR-GRAPH$(G - e, X, S)$
6     ENUMERATE-FOR-GRAPH$(G, X \cup \{e\}, S)$
```

Lemma 3.12. *Let S be a maximal well-ordered k-degenerate graph with n vertices and let $G \neq S$ be a subgraph of S with n vertices. Then G is a proper subgraph of S if and only if $E(S) \setminus E(G) \subseteq F(S)$. That is, the proper subgraphs of S are exactly the graphs*

$$P(S) = \{S - E' \mid E' \subseteq F(S),\ S - E' \text{ is well-ordered (strongly) } k\text{-degenerate}\} . \tag{3.18}$$

Proof. Let G be a k-degenerate graph. First, assume $E(S) \setminus E(G) \subseteq F(S)$. Then the edges which must be added to G in order to obtain S are exactly the lexicographically smallest missing edges, which are exactly the edges in $X(G)$.

Second, assume that G is a proper subgraph of S, that is, by adding the edges in $X(G)$ we obtain S. By definition $X(G)$ contains, for each vertex v_i of G, $\mathrm{d}^+_{\max}(v_i) - \mathrm{d}^+(v_i)$ the lexicographically smallest edges not in $E(G)$. Therefore $X(G) \subseteq F(S)$.

To see why Equation (3.18) holds, consider a set $E' \subseteq F(S)$ and let $G := S - E'$. Then

$$E(S) \setminus E(G) = E(S) \setminus (E(S) \setminus E') = E' \subseteq F(S) ,$$

that is, G is a proper subgraph of S. On the other hand, if G is a proper subgraph of S, then $E' := E(S) \setminus E(G) \subseteq F(S)$ and $G = S - E'$. □

The characterization of the proper subgraphs according to Equation (3.18) can be used to design an efficient enumeration algorithm for well-ordered (strongly) k-degenerate graphs. The following Algorithm 3.9 enumerates all well-ordered k-degenerate graphs using this characterization by branching on the subsets of the foundation. The input of the algorithm consists of a graph G, a set of edges $X \subseteq E(G)$ that may not be removed from G and a maximal well-ordered k-degenerate graph S. At each step the algorithm branches on some edge $e \in (F(S) \cap E(G)) \setminus X$. We call the set of edges X *forced* since they are not used for the branching. Initially, none of the edges is forced. For each edge $e \in (F(S) \cap E(G)) \setminus X$ the algorithm checks if $G - e$ is well-ordered (strongly) k-degenerate. If this is the case, the algorithm outputs $G - e$ and recursively calls itself on the smaller graph. If the graph is not well-ordered (strongly) k-degenerate, then no subgraph of G with n vertices can be well-ordered (strongly) k-degenerate. In this case we need not branch further.

Lemma 3.13. *Let S be a maximal well-ordered k-degenerate supergraph of G. Then Algorithm 3.9 enumerates and prints all proper well-ordered (strongly) k-degenerate subgraphs of S in amortized time $\mathcal{O}(nm + m^2)$ per printed graph on input $(S, \emptyset, S)$.*

```
Algorithm 3.10: ENUMERATE-DEGENERATE($n[, m]$)
  Input: number of vertices $n$
  Output: all well-ordered (strongly) $k$-degenerate graphs
1 for all maximal well-ordered $k$-degenerate graphs $S$ do
2     print $S$
3     ENUMERATE-FOR-GRAPH($S, \emptyset, S[, m]$)
```

Proof. The algorithm recursively branches on the edges of $F(S)$ enumerating the subsets of $F(S)$ with decreasing size. Clearly, the algorithm only prints proper subgraphs of S by Equation (3.18).

On the other hand, let G be a proper well-ordered strongly k-degenerate subgraph of S, that is, $E(S) \setminus E(G) \subseteq F(S)$. Then we need to show that G is enumerated by the algorithm. To see this, consider a arbitrary set of edges $Y \subseteq E(S) \setminus E(G)$. Since both G and S are well-ordered strongly k-degenerate, so is $G + Y$. Hence, for any removal order of the edges $e_1, \ldots, e_\ell$ such that $\{e_1, \ldots, e_\ell\} = E(S) \setminus E(G)$ the graph $G_i := G + \{e_1, \ldots, e_i\}$ is well-ordered k-degenerate for each $1 \leq i \leq \ell$. Therefore the algorithm reaches and prints G when branching on the edges in $F(S)$.

Further, each well-ordered (strongly) k-degenerate graph is printed only once. To see this, note that the printing of $G - e$ can be associated with the removal of the edge e. However, any edge is either in the set X of forced edges and is not removed any more, or it is removed at most once without re-insertion, since the algorithm recurses on $G - e$.

Next, we show that the amortized time per generated graph is bounded by $\mathcal{O}(nm + m^2)$. First note, that each single call of the algorithm (without recursion) can be performed in $\mathcal{O}(n + m)$ time. Consider the recursion tree T, in which each node corresponds to a call of the algorithm. Each node in T that corresponds to a printed graph G is called a *print-node*. Let G be a graph printed by the algorithm and let $T_{(G,X,S)}$ denote the subtree rooted in the print-node v_G corresponding to a call of the algorithm with arguments G, X and S. We charge v_G with all nodes in $T_{(G,X,S)}$ that can be reached from v_G in T without crossing any other print-node. Each call to the recursion in Line 5 is only performed if $G - e$ is well-ordered (strongly) k-degenerate. Hence, $G - e$ will be charged for this call. Therefore, G will only be charged for calls in Line 6. The same holds for any call involving G and a set $X' \supset X$. Hence, G is charged for at most m calls of the algorithm, which yields an amortized running time bounded by $\mathcal{O}(nm + m^2)$ per graph. □

Finally, Algorithm 3.10 contains the pseudo-code for the topmost level. Note that the algorithm can be used to enumerate both regular and well-ordered strongly k-degenerate graphs with n vertices.

Theorem 3.11. *Algorithm 3.10 enumerates all well-ordered (strongly) k-degenerate graphs in amortized $\mathcal{O}(nm + m^2)$ time per graph.*

Proof. Since the algorithm enumerates the proper subgraphs of all maximal well-ordered k-degenerate graphs it enumerates every graph exactly once. By Equation (3.18) and correctness of Algorithm 3.9 (Lemma 3.13) the algorithm enumerates all proper subgraphs for a given maximal well-ordered k-degenerate graph. Since we can generate and print the maximal well-ordered k-degenerate graphs in amortized linear time according to Theorem 3.10, the overall running time of the algorithm is $\mathcal{O}(nm + m^2)$ per enumerated and printed graph. $\square$

3.7.2 Enumerating Well-Ordered Degenerate Graphs with n Vertices and m Edges

Next, we describe how to enumerate well-ordered (strongly) k-degenerate graphs with n vertices and m edges. In order to obtain an efficient algorithm we first establish a simple criterion to decide if a given graph G contains a subgraph H such that $G - E(H)$ is strongly k-degenerate. Let G be a given well-ordered strongly k-degenerate graph with m edges and let $X \subseteq E(G)$ be a set of edges. As in the previous section, we would like to branch, for given graph G and a set X of forced edges, on the set of edges in $(E(G) \setminus X) \cap F(S(G))$. That is, for a given graph G and a set $X \subseteq E(G)$, we would like to decide, whether G contains a proper well-ordered (strongly) k-degenerate subgraph G' of $S(G)$ such that G' contains all edges in X. We show that this problem is equivalent to computing a generalized matching in a properly defined graph.

Let G_X^F be the graph induced by the edges in $(E(G) \setminus X) \cap F(S(G))$. That is, G_X^F contains all edges that may be removed from G in order to obtain G' with the desired properties. Further, we define $u_i = \mathrm{d}_G(v_i) - k$ for $i = 1, \dots, n$. Hence, u_i is the maximum number of edges incident to v_i that we may remove from G such that the resulting degree of v_i is at least k. Let H be a subgraph of G_X^F with the maximum number of edges such that $\mathrm{d}_H(v_i) \le u_i$ for all i. We call the maximum number of possible edges of H the *excess* of G with respect to X, denoted by $\mathrm{xs}(G, X)$. The excess can be computed in time $\mathcal{O}(\sum_{i=1}^{n} \sqrt{u_i} \cdot m) \subseteq \mathcal{O}(n^{3/2} m)$ by computing a generalized matching using an algorithm by Gabow [Gab83]. The following lemma shows that G contains a subgraph G' with m' edges and the desired properties if and only the excess of G with respect to X is at least $m - m'$, that is, if we are allowed to remove a sufficiently large number of edges without violating the lower bound for the degrees.

Lemma 3.14. *Let $G = (V, E(G))$ be a well-ordered strongly k-degenerate graph with n vertices and $m > m'$ edges and let $X \subseteq E(G)$ be a set of edges. Then G contains a well-ordered strongly k-degenerate subgraph G' with n vertices and m' edges that is a proper subgraph of $S(G)$ and that contains all edges in X if and only if* $\mathrm{xs}(G, X) \ge m - m'$.

Proof. “if”: Assume that $\mathrm{xs}(G, X) \ge m - m'$. Then G contains a subgraph H with edge-set $E(H)$ such that $|E(H)| \ge m - m'$ and $E(H) \subseteq (E(G) \setminus X) \cap F(S(G))$. Let $E' \subseteq E(H)$ with $|E'| = m - m'$. Then $G - E'$ is a well-ordered strongly k-degenerate proper subgraph of $S(G)$. Clearly, $G - E'$ is strongly k-degenerate since all vertices have degree $\ge k$ by definition of u_i. Additionally, $G - E'$ is a proper subgraph of $S(G)$ since $E' \subseteq F(S(G))$ by Lemma 3.12.

“only if”: Assume that G contains a well-ordered strongly k-degenerate subgraph G' with m' edges which is a proper subgraph of $S(G)$ and which contains all edges in X. Let $E' := E(G) \setminus E(G')$. Then clearly, $E' \subseteq E(G) \setminus X$ and $E' \subseteq F(S(G))$ by Lemma 3.12, that is, $E' \subseteq (E(G) \setminus X) \cap F(S(G))$. Hence, E' is a subset of $E(G_X^F)$. Let $H := G(V, E')$, then H is a subgraph of G_X^F and for all vertices v_i we have $\mathrm{d}_H(v_i) \le u_i = \mathrm{d}_G(v_i) - k$ and, therefore, $\mathrm{xs}(G, X) \ge |E'| = m - m'$. $\square$

Using Lemma 3.14 in combination with Algorithms 3.10 and 3.9 we can enumerate all well-ordered strongly k-degenerate graphs with n vertices and m edges by slightly modifying

Algorithm 3.11: Enumerate-For-Graph(G, X, S, m)

```
Input: (well-ordered strongly) $k$-degenerate graph $G$, forced edges $X$, super-graph $S$
Output: all well-ordered (strongly) $k$-degenerate proper subgraphs of $S$ containing no
        edges in $X$
1 if $(F(S) \cap E(G)) \setminus X \neq \emptyset$ then
2     $e \leftarrow$ choose an edge in $(F(S) \cap E(G)) \setminus X$
3     if $G - e$ is (strongly) $k$-degenerate and $\mathrm{xs}(G - e, X) \geq |E(G - e)| - m$ then
4         if $|E(G)| = m$ then
5             print $G - e$
6         Enumerate-For-Graph($G - e, X, S$)
7     if $\mathrm{xs}(G, X \cup \{e\}) \geq |E(G)| - m$ then
8         Enumerate-For-Graph($G, X \cup \{e\}, S$)
```

Algorithm 3.9: We extend the check in Line 3 by testing if $\mathrm{xs}(G - e, X) \geq |E(G)| - m$. We further add a similar check in Line 6. The pseudo-code is listed in Algorithm 3.11.

Theorem 3.12. *Algorithm 3.11 enumerates and prints all well-ordered (strongly) k-degenerate graphs with n vertices and m edges in amortized $\mathcal{O}(n^{3/2}m^2)$ time per printed graph.*

Proof. The proof is similar to the proof of Theorem 3.11. The running time results from the fact that we can check $\mathrm{xs}(G, X) \geq i$ in time $\mathcal{O}(n^{3/2}m)$. Note, that the algorithm recurses only if the subtree rooted in the current node of the recursion tree contains at least one more print-node. Let G be a well-ordered (strongly) k-degenerate graph that is printed by the algorithm. In the search tree we then charge the node v_G associated with G with all nodes on the path from the source of the recursion tree to v_G. Since the algorithm branches only if the corresponding subtree of the recursion tree contains a node that can be associated with a printed graph, all nodes in the recursion tree are charged to a printed graph. However, since the height of the recursion tree is bounded by m, each printed graph is charged at most m nodes of the recursion tree. Thus, the running time is bounded by $\mathcal{O}(n^{3/2}m^2)$ per printed graph. □

In order to enumerate ordinary well-ordered k-degenerate graphs with n vertices and m edges we need only check if $|E(G - e) \cap F(S(G))| \geq |E(G - e)| - m$. This can be done in constant time if we allow to have a marker for edges in the foundation of $S(G)$. Testing whether the graph is well-ordered k-degenerate and printing the graph can be done in time $\mathcal{O}(n + m)$. Therefore, we can enumerate ordinary well-ordered k-degenerate graphs with n vertices and m edges in time $\mathcal{O}(nm + m^2)$ per enumerated graph.

3.8 Concluding Remarks

In this chapter we studied the problem of enumerating and generating well-ordered k-degenerate graphs. On the one hand, we presented efficient algorithms for generating well-ordered k-degenerate graphs uniformly at random whose running times are almost optimal in the real-RAM machine model, given that we are allowed a polynomial amount of time and space for a precomputation. The precomputation is completely amortized if a linear number of graphs must be generated for the same set of parameters.

However, since the numerical computations involved in these algorithms are rather costly and the numbers that are involved become huge even for small-sized graphs, we additionally presented fast and easy-to-implement algorithms whose running time is linear in the size of the generated graphs in the classical RAM machine model. For well-ordered k-degenerate graphs with given number of vertices we presented a fast algorithm based on an approximation of the uniform distribution. Further, we studied strongly k-degenerate graphs, a natural generalization of k-regular graphs and we presented fast and efficient algorithms for generating graphs from this class. The presented algorithms can be used to generate k-regular graphs without modification and thus yield an interesting new approach to generating regular graphs.

While the class of well-ordered k-degenerate graphs is only a subset of the labeled k-degenerate graphs, our approach is preferable to the classical approach of generating labeled graphs as we elaborated at the beginning of this chapter, since it allows to filter out some isomorphies resulting in generators whose distribution is closer to the ideal uniform distribution on the unlabeled k-degenerate graphs. To the best of our knowledge, this is the first application of a non-standard labeling scheme.

On the other hand, we studied the problem of enumerating well-ordered k-degenerate graphs and we presented algorithms for enumerating k-degenerate and well-ordered k-degenerate graphs, respectively. Again, these algorithms can equally be used to enumerate regular graphs.

Open problems One of the major open problem in this chapter concerns the problem of generating strongly k-degenerate graphs uniformly at random. While we presented an algorithm for generating these graphs uniformly at random based on rejection, the efficiency of this approach is far from being satisfactory. Thus, the problem of generating strongly well-ordered k-degenerate graphs remains widely open.

Although we presented fast uniform as well as fast and easy-to-implement complete generators for well-ordered strongly k-degenerate graphs, it is not clear what the distribution of the presented non-uniform generators is. Thus, it is an open problem to study this distribution or to provide different fast and easy-to-implement algorithms for generating these classes of graphs while providing approximation guarantees on the distribution of these generators, thus closing the gap between theory and practice a little further. As a preliminary step it may also be interesting to consider 2-degenerate or 3-degenerate graphs.

Finally, it seems to be worthwhile to study new labeling schemes for various classes of graphs in order to design new generators for labeled graphs whose distribution is closer to the uniform distribution than the distribution of generators relying on the classical labeling scheme.

Chapter 4

Optimal Routing Cost Tree Augmentation

The performance or efficiency of communication and transportation networks is frequently assessed in terms of its routing cost. This quantity is defined as the (weighted) sum of the lengths of the shortest paths between all pairs of nodes of the network. On the other hand, the cost of constructing or maintaining the network is often considered to be proportional to the number of links in the network. Therefore, tree networks have received considerable attention in network construction.

We study two augmentation problems on trees whose goal is to minimize the resulting routing cost. First, we study the problem of finding the optimal connection between two disconnected vertex-weighted trees. We are given a distance function on the vertices and seek to minimize the routing cost of the tree resulting from adding one single edge between the two trees. The problem arises, for instance, when augmenting and/or repairing communication networks or infrastructure networks. We present an asymptotically optimal quadratic-time algorithm for the general case and show that the problem can be solved more efficiently for the Euclidean metric, when vertices are mapped to points in the plane, as well as for compactly representable graph metrics.

Second, we study the problem of inserting an additional shortcut into a vertex-weighted directed tree network such the weighted routing cost of the resulting network is minimized. While this problem can be solved in quadratic time by exhaustive search, we show that it be solved in linear time in a directed path and a star-shaped network and that it can be solved in $\mathcal{O}(n \log n)$ time in an arbitrary directed tree network with n nodes. This chapter is based on joint work with Mong-Jen Kao, Bastian Katz, Rolf Klein, Elmar Langetepe, Der-Tsai Lee, Martin Nöllenburg, Ignaz Rutter and Dorothea Wagner [KKK$^+$11a, KKL$^+$11].

4.1 Introduction

In the construction of communication and infrastructure networks we often have to find a reasonable balance between the cost for establishing the links between the vertices in the network and the performance of the network in terms of various quality measures, such as routing cost, connectivity and diameter. While the cost should be minimized and increases with each established link, the performance of the network should be maximized and typically improves when more links are added. This tradeoff can be formalized in different ways, for instance, by modeling the network construction problem as a multi-objective optimization problem. However, motivated by practical applications of this problem it is quite common to assume that we are given a limited budget for the construction cost and wish to optimize the performance of the network subject to this constraint.

The *Optimal Network Problem*, which has been introduced by Scott [Sco69], addresses the problem of optimizing the *routing cost* of a network, defined by the sum of the shortest paths between all pairs of vertices in the graph. Due to its importance for communication networks, this problem has received considerable attention, among others by Dionne and Florian [DF79] and Wong [Won80] and, more recently, by Fischetti et al. [FLS02].

If the budget for establishing the links in a network is rather tight, a tree is often the only affordable infrastructure and the construction of tree networks therefore gained considerable attention. However, Johnson et al. [JLK78] prove that the optimal network problem is NP-complete, even if all edges have the same length and the network must be a tree. This problem is also called the *Minimum Routing Cost Spanning Tree Problem (MRCST)*. More recently, Wu et al. presented an FPTAS for this problem [WLB$^+$99] and Fischetti et al. [FLS02] studied exact algorithms for computing the minimum routing cost spanning tree.

In this chapter we consider two problems related to the augmentation of tree networks. The first problem we consider is the problem of connecting a disconnected tree network consisting of two vertex-weighted trees by inserting an additional edge such the weighted routing cost of the resulting tree is minimized. The second problem is to augment a given vertex-weighted tree network by inserting an additional edge, called a shortcut, such that the weighted routing cost of the resulting graph is minimized. While both problems can be solved by exhaustive enumeration of the possible edges in quadratic time with respect to the number of vertices in the trees, we present optimal or almost optimal algorithms.

Related problems In almost all areas related to network construction, special attention has been paid to trees. A comprehensive overview of network construction problems involving trees can be found in the textbook by Wu and Chao [WC04]. Graph augmentation problems have been widely studied in computer science. Typical problems involve augmenting the connectivity of networks in order make them more robust against failure of links. Graph augmentation problems in network design are surveyed by Frank [Fra94] and more recently by Nagamochi [Nag00]. In addition to the classical graph augmentation problem, several variants of the problem have been studied. For instance, Nutov [Nut09] presents approximation algorithms for the problem of augmenting graphs with the minimum number of edges such that, for each pair of vertices u and v in the resulting network, there is at least a given number of edge-disjoint paths, that additionally do not share any vertex from a given set of vertices S. Further, Ishii et al. [IAN10] consider the problem of augmenting a given graph with respect to its edge-connectivity between single vertices and given sets of vertices in the graph, called areas, and show that the problem is polynomial-time solvable if the network is required to be at least k-connected with $k \geq 3$.

Contribution In Section 4.2 we consider the problem of augmenting two disconnected trees with n_1 and n_2 vertices, respectively. First, we consider general distance functions in Section 4.2.1. We show that both the optimal routing cost augmentation problem and the optimal routing cost replacement problem can be solved in $\Theta(n_1 \cdot n_2)$ time, which is optimal. Second, we assume that vertices are points in the plane and that the distance between points is equal to the Euclidean distance in Section 4.2.2. We show that both the augmentation problem and the replacement problem can be solved more efficiently in $\mathcal{O}(n \log \min\{n_1, n_2\})$ time by querying an additively weighted Voronoi diagram of a suitably chosen set of points.

Finally, we adapt this idea to general graph metrics by computing an additively weighted Voronoi diagram on graphs in Section 4.2.3. This yields an $\mathcal{O}(n \log n)$-time algorithm for compactly representable metrics, that is, metrics that are representable as sparse graphs.

In Section 4.3, we consider the problem of augmenting a connected tree-network by an additional edge reducing the routing cost. Subsequently, we show how this problem can be reduced to computing the upper envelope of an arrangement of piecewise linear functions in Section 4.3.1 and prove that this problem can be solved in linear time in Section 4.3.2. Finally, we show how general trees can be reduced to the special case of paths in Section 4.3.3.

4.2 Connecting Two Trees

In this section, we consider the problem of augmenting two disconnected, vertex-weighted tree networks by a single edge such that the resulting network connected and the weighted routing cost is minimized.

More formally, we consider the following problem. We are given a set of vertices V as well as some distance function d on V such that $d(v,v) = 0$ and $d(u,v) = d(v,u) \geq 0$ for all vertices $u, v \in V$. Further, we are given a partition of $V = V_1 \cup V_2$ and two disjoint trees $T_1 = (V_1, E_1)$ and $T_2 = (V_2, E_2)$ on V_1 and V_2, respectively. We write $n = |V|$, $m = |E|$, $n_i = |V_i|$ and $m_i = |E_i|$ for $i \in \{1,2\}$. For each tree T on a subset $V' \subseteq V$, we consider the tree metric d_T, which is defined on V' such that the distance between $u, v \in V'$ is equal to the sum of the distances on the uniquely defined path between u and v. Further, we assume each vertex $v \in V$ has some non-negative demand $c(v)$. For $V' \subseteq V$ we write $c(V') := \sum_{v \in V'} c(v)$ as a shorthand. We define the *weighted routing cost* of T as

$$\mathrm{rc}(T) = \sum_{(u,v) \in V \times V} c(u) \cdot c(v) \cdot d_T(u,v) \; .$$

The demands can be considered to be an indicator for the importance of the vertices in the network. The amount of traffic between two vertices in the network is scaled by the product of the demands modeling the fact that important vertices are usually involved in more traffic than less important vertices and that the traffic between two important vertices is usually larger than that between an important and a less important vertex.

The Optimal Routing Cost Augmentation Problem is to find vertices $u \in V_1$ and $v \in V_2$ such that the routing cost of the tree $T_{uv} = (V, E_1 \cup E_2 \cup \{uv\})$ is minimized as illustrated in Figure 4.1, for instance.

For the Optimal Routing Cost Replacement Problem we are additionally given a pair of vertices $u \in V_1$ and $v \in V_2$ that should be excluded from the solution (since the corresponding edge must be replaced). We can solve this problem by simultaneously computing the best and second-best solution. If the best solution coincides with uv, then we return the second-best solution.

4.2.1 An Optimal Algorithm for the General Case

In this section, we consider general distance functions on the vertex set. We show that the problem can be solved in $\Theta(n_1 \cdot n_2)$ time, which is optimal. For ease of notation we write $C_1 = c(V_1)$ and $C_2 = c(V_2)$ for the total demand in T_1 and T_2, respectively. Given two

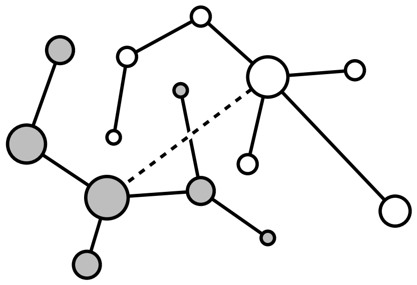


Figure 4.1: Two vertex-weighted trees and the best connection with respect to routing cost (dashed).

vertices $u \in V_1$ and $v \in V_2$, the routing cost of the tree T_{uv} resulting from joining T_1 and T_2 by the edge uv is given by

$$\begin{aligned} rc(T_{uv}) &= rc(T_1) + rc(T_2) \\ &\quad + C_2 \cdot \sum_{u' \in V_1} c(u') \cdot d_{T_1}(u', u) \\ &\quad + C_1 \cdot \sum_{v' \in V_2} c(v') \cdot d_{T_2}(v, v') \\ &\quad + C_1 \cdot C_2 \cdot d(u, v) \ . \end{aligned} \tag{4.1}$$

It is composed of the routing cost inside the subtrees T_1 and T_2 of T_{uv}, respectively, and the routing cost effected by the shortest paths using the edge uv between the two trees. Since the total sum of demands for these paths equals $C_1 \cdot C_2$, the edge uv contributes a total amount of $C_1 \cdot C_2 \cdot d(u, v)$ to the routing cost. Furthermore, each shortest path starting at u' in T_1 and ending at u can be extended to a shortest path ending at some vertex v' in T_2. Hence, each shortest path of this kind contributes its length, weighted by its demand $c(u')$ and the total sum of the demands C_2 in T_2, to the routing cost. The situation is symmetrical for the paths starting in T_2 and ending at v.

Since the routing costs of T_1 and T_2 do not depend on the choice of the link between the two trees, our problem is equivalent to minimizing the remaining summands in equation (4.1).

We define the weight of a vertex $u \in V_1$, denoted by $w(u)$, as the sum of lengths of all shortest paths starting at $u' \in V_1$ and ending at u, weighted by the demand of u', that is,

$$w(u) = \sum_{u' \in V_1} c(u') \cdot d_{T_1}(u', u) \ .$$

We define the weight of a vertex $v \in V_2$, denoted by $w(v)$, analogously. Hence, we seek to minimize the term

$$rc'(T_{uv}) = C_2 w(u) + C_1 w(v) + C_1 \cdot C_2 \cdot d(u, v) \tag{4.2}$$

over all possible combinations of $u \in V_1$ and $v \in V_2$.

The weights of the trees can be computed in linear time as follows. First we compute the total demands in T_1 and T_2, respectively. We compute the weights in T_1 by rooting the tree in some vertex r and performing one bottom-up pass over the tree, followed by a top-down pass. For a vertex u in T_1 we denote the subtree rooted in u by T_u.

In the bottom-up pass, we compute two values for each vertex $u \in V_1$: the total demand $\gamma(u)$ of the vertices in T_u, and the sum $\lambda(u)$ of the shortest paths starting at some vertex u' in T_u and ending at u, weighted by the demand of u', that is,

$$\gamma(u) = \sum_{u' \in V(T_u)} c(u')$$

and

$$\lambda(u) = \sum_{u' \in V(T_u)} c(u') d(u', u) \; .$$

For a vertex u with children $u_1, \dots, u_k$ these values can be computed in linear time as

$$\gamma(u) = c(u) + \sum_{i=1}^{k} \gamma(u_i)$$

and

$$\lambda(u) = \sum_{i=1}^{k} \big(\lambda(u_i) + \gamma(u_i) \cdot d(u_i, u)\big) \; ,$$

respectively. In the top-down pass, we compute the weight for each vertex $v \in V_1$. For the root r this weight is equal to $\lambda(r)$. For a vertex v with father $u \in V_1$ the weight can be computed by

$$w(v) = w(u) + (C_1 - 2\gamma(v)) d(u, v) \; .$$

This equation is due to the fact that the weight of v is obtained from the weight of u by removing the demand $\gamma(v)$ in the subtree of v from the edge uv and adding the remaining demand $C_1 - \gamma(v)$ to the edge uv. For T_2 we proceed analogously.

Having this, we can compute the best and second-best connection between the two trees by enumerating all possible pairs uv such that $u \in V_1$ and $v \in V_2$, which yields a total running time of $\mathcal{O}(n_1 \cdot n_2)$. Note, that the described algorithm only finds the best or second-best solution, but does not compute the routing cost of this solution. If we have no restriction on the distance between the vertices, however, the algorithm is optimal.

Theorem 4.1. *The optimal routing cost augmentation problem and the optimal routing cost replacement problem can be solved in $\mathcal{O}(n_1 \cdot n_2)$ time for general distance function. This is optimal in the algebraic decision tree model.*

Proof. We have already outlined the algorithm and argued why it runs within the stated time complexity. It remains to show the lower bound on the running time. For this, we assume that we are given a set of integers $a_1, \dots, a_N$. We construct an instance of the optimal routing cost augmentation problem such that finding the minimum routing cost connection between the two trees is equivalent to the minimum of the numbers $a_1, \dots, a_N$. For this problem, we need at least $N - 1$ comparisons in the algebraic decision tree model of computation.

Let $N = n_1 n_2$ be any factorization of N and let V be a set of $n_1 + n_2$ vertices. Further, let $V_1, V_2 \subseteq V$ be a partition of V such that $|V_1| = n_1$ and $|V_2| = n_2$ and let T_1 and T_2 be two arbitrary trees on V_1 and V_2, respectively. We set the distance between two vertices in the same tree equal to one. Let $x : V_1 \times V_2 \to \{a_1, \ldots, a_N\}$ be a bijective mapping between the pairs of vertices in V_1 and V_2 and the numbers a_i. Then we choose the remaining distances as follows. Let W_1 and W_2 be the maximum weights of the vertices in T_1 and T_2, respectively. For $u \in V_1$ and $v \in V_2$ we define

$$d_0(u,v) = C_2 W_1 + C_1 W_2 - C_2 w(u) - C_1 w(v) \ .$$

Further, we set

$$d(u,v) = \frac{d_0(u,v) + x(u,v)}{C_1 C_2} \ .$$

Then $\mathrm{rc}'(T_{uv}) = C_2 W_1 + C_1 W_2 + x(u,v)$. For both the augmentation and the replacement problem we need to compute the minimum routing cost solution. However, minimizing the routing cost for the given instance is equivalent to computing the minimum over the values $x(u,v)$ for $u \in V_1$ and $v \in V_2$. Hence, in the algebraic decision tree model of computation, we need at least $n_1 \cdot n_2 - 1$ comparisons, which completes the proof. □

4.2.2 An Efficient Algorithm for the Euclidean Metric

The proof for the lower bound in the previous section crucially exploits the fact that we can choose distances between the vertices in an arbitrary fashion. If this is not the case, we can come up with more efficient algorithms.

In this section we consider the case that vertices are points in the plane and that the considered metric d is the Euclidean metric. In this case, we can compute the best connection between two trees in $\mathcal{O}((n_1 + n_2) \log \min\{n_1, n_2\})$ time. Throughout the section, we do not distinguish between vertices and points.

Theorem 4.2. *The optimal augmentation problem for the Euclidean metric can be solved in $\mathcal{O}((n_1 + n_2) \log \min\{n_1, n_2\})$ time.*

Proof. Without loss of generality we may assume that $n_2 \leq n_1$. Let $\sigma : \mathbb{R}^2 \to \mathbb{R}^2$ be an isotropic scaling with scale factor $s = C_1 \cdot C_2$, that is, σ scales distances by a factor s and we thus have

$$d(\sigma u, \sigma v) = C_1 \cdot C_2 \cdot d(u,v) \ .$$

Let σV_1 and σV_2 denote the scaled sets of points.

For $x \in \mathbb{R}^2$ and $\widetilde{v} \in \sigma V_2$ we define a new distance function, defined by $d_+(x, \widetilde{v}) := d(x, \widetilde{v}) + C_1 \cdot w(v)$, where w is defined as in the previous section. The *additively weighted Voronoi cell of* $\widetilde{v}$ is the locus of points

$$\{x \in \mathbb{R}^2 \mid \forall \widetilde{u} \in \sigma V_2 \setminus \{\widetilde{v}\} : d_+(x, \widetilde{v}) < d_+(x, \widetilde{u})\}$$

The additively weighted Voronoi diagram $\mathcal{V}$ defined by d_+ consists of the additively weighted Voronoi cells of the points in σV_2 and can be computed in $\mathcal{O}(n_2 \log n_2)$ time [For87].

For each point $u \in V_1$, we locate the nearest neighbor σv of σu in $\mathcal{V}$ using an algorithm with $\mathcal{O}(\log n_2)$ query time described by Kirkpatrick [Kir83]. Then σv satisfies

$$d_+(\sigma u, \sigma v) = \min_{v' \in V_2} d_+(\sigma u, \sigma v')$$

and we have

$$\begin{aligned} d_+(\sigma u, \sigma v) &= d(\sigma u, \sigma v) + C_1 \cdot w(v) \\ &= C_1 \cdot C_2 \cdot d(u, v) + C_1 \cdot w(v) \ . \end{aligned}$$

Hence, $v \in V_2$ is the best endpoint of an edge starting at $u \in V_1$ with respect to routing cost. Minimizing $C_2 \cdot w(u) + d_+(\sigma u, \sigma v)$ over all vertices $\sigma u \in V_1$ and their respective nearest neighbor $\sigma v \in V_2$ will thus minimize the overall routing cost. The resulting overall running time is $\mathcal{O}(n_1 \log n_2 + n_2 \log n_2)$. □

In order to solve the replacement problem, we also need to compute the second-best solution. We can do this as follows. Let $u^* \in V_1$ and $v^* \in V_2$ be the best solution computed by the algorithm above. This algorithm can trivially be modified to simultaneously compute

$$\min_{u \in V_1 \setminus \{u^*\}, v \in V_2} \mathrm{rc}'(T_{uv})$$

in the same time complexity. By additionally computing the Voronoi diagram only for the points in $V_2 \setminus \{v^*\}$ and repeating the algorithm on this instance, we can also compute

$$\min_{u \in V_1, v \in V_2 \setminus \{v^*\}} \mathrm{rc}'(T_{uv}) \ .$$

Clearly, the second-best solution is either of the two. Hence, we have the following corollary.

Corollary 4.1. *The optimal routing cost replacement problem for the Euclidean metric can be solved in time $\mathcal{O}((n_1 + n_2) \log n_2)$.*

Note that the same approach can also be used in a planar setting, that is, when the newly introduced edge connecting the two trees may not intersect any other edge of the two trees. In this case we compute an additively weighted constrained Voronoi diagram, which can be done by adapting Fortune's sweepline algorithm [For87] with $\mathcal{O}(n \log n)$ running time. In a constrained Voronoi diagram, we are given an additional set of line segments representing obstacles. Whenever the straight line connecting two points intersects one of the obstacles, the distance between the two points is assumed to be infinity, otherwise, it is equal to the (weighted) Euclidean distance between the points. In our application each edge defined by one of the trees is one such obstacle. Seidel shows how to adapt Fortune's algorithm to compute the constrained Voronoi diagram [Sei88]. The adaption to additively weighted sites has been sketched in Fortune's original paper [For87].

Corollary 4.2. *The planar augmentation problem for the Euclidean metric can be solved in $\mathcal{O}((n_1 + n_2) \log n_2)$ time.*

4.2.3 General Metrics

Every finite metric d can be encoded by a finite graph $M = (V, D)$ where each edge $e \in D$ has some length $\ell(e)$ and the distance d between two vertices in V is equal to the sum of the lengths of the shortest path between the vertices in the graph in terms of the edge lengths. We can directly translate our idea from the previous section to this setting by computing the additively weighted Voronoi diagram in M instead. Although the computation of various

Voronoi diagrams on graphs has been considered by Hurtado et al. [HKLS04], among them a multiplicatively weighted Voronoi diagram, we are not aware of any investigation of the additively weighted Voronoi diagram on graphs. The following theorem is similar to the results by Hurtado et al. [HKLS04]. We assume that the additively weighted Voronoi diagram of a set of sites $S \subseteq V$ on a metric graph $G = (V, E)$ is completely known if every vertex $v \in V \setminus S$ knows its nearest neighbor in S and we know the bisector point for each edge, if it exists.

Theorem 4.3. *The additively weighted Voronoi diagram of a set of sites $S \subseteq V$ on a graph $G = (V, E)$ has complexity $\Theta(m)$ and can be computed in time $\mathcal{O}(m + n \log n)$.*

Proof. Each edge of the graph contains at most one bisector point, since moving along the edge will alter the additively weighted distances by the same amount—either increasing or decreasing—for all distances. Hence we have at most m bisector points. On the other hand, we can have exactly m bisectors by setting $V' = V$. Hence, the complexity of the additively weighted Voronoi diagram is $\Theta(m)$.

To compute the additively weighted Voronoi diagram in G we use the parallel Dijkstra algorithm proposed by Erwig [Erw00] with running time $\mathcal{O}(m + n \log n)$. To compute the diagram, we run Dijkstra's algorithm in parallel using the vertices in S as starting points. For a vertex $v \in V \setminus S$ and some vertex $s \in S$ the distance between v and s is $d_s(v, s) = d_G(v, s) + w(s)$. Whenever a vertex $v \in V \setminus S$ is settled, we update its closest neighbor in S and we annotate v with the distance to its closest neighbor. The bisector points can be computed in $\mathcal{O}(m)$ time from this information as follows. For each edge uv we obtain the closest sites s_u and s_v in S that have been stored at the vertices. If $s_u \neq s_v$, then the edge uv contains a bisector point. This point is located halfway between s_u and s_v on uv and can be computed in a straightforward way from the distances d_u and d_v from u and v to s_u and s_v, respectively. □

Using this result, we can almost directly translate the technique for the Euclidean case to the general metric case studied in this section.

Theorem 4.4. *The optimal routing cost augmentation problem for general metrics can be solved in time $\mathcal{O}(m + n \log n)$ if the metric is given by a graph $M = (V, D)$ with edge length function ℓ.*

Proof. Instead of scaling the point set as in the Euclidean case, we scale the lengths of the edges in G by a factor $C_1 C_2$, that is, instead of using ℓ to assess the distance between two vertices in M, we use $C_1 C_2 \ell$. The rest of the proof is completely analogous. We compute the additively weighted Voronoi diagram on M for the set of sites V_2 in $\mathcal{O}(m+n \log n)$ as described in Theorem 4.3. Then we locate the vertex $u \in V_1$ that minimizes $C_2 \cdot w(u) + d_+(u, v)$ where $d_+(u, v)$ is the scaled and additively weighted distance between u and its closest neighbor v. The resulting time complexity is $\mathcal{O}(m + n \log n)$ according to Theorem 4.3. □

Again we can proceed as in the Euclidean case in order to compute the second-best connection between the two trees.

Corollary 4.3. *The optimal routing cost replacement problem for general metrics can be solved in time $\mathcal{O}(m + n \log n)$ if the metric is given by a graph $M = (V, D)$ with edge length function ℓ.*

Although this result does not provide an asymptotic improvement in the worst-case, it does show that we can efficiently solve the augmentation problem for compactly representable metrics. If the graph representing the metric is sparse, then the above theorem states that we can solve the augmentation problem in $\mathcal{O}(n \log n)$ as in the Euclidean case.

4.3 Optimal Shortcuts in Trees

In this section, we are interested in augmenting a given network by adding an extra edge to it in such a way that its routing cost is reduced as much as possible. In contrast to the previous section, we assume that each edge is of unit length, reflecting a situation where long distance traffic in transportation networks is fast, while changing between highways or railways requires time consuming inner-city travel and waiting, respectively. Similarly, this model reflects a situation encountered in communication networks, where passing a message along a link is cheap, while the processing and distribution of messages at the nodes of the network requires expensive computational work. Further, we consider the special case in which the network is a tree.

A similar augmenting problem for general graphs has been studied by Farshi et al. [FGG08] with respect to the dilation (or: stretch factor, spanning ratio) of a network. Given a set of vertices V in the Euclidean plane and a network $G = (V, E)$ interconnecting these vertices, the dilation of the network is defined as the maximum of the quotient $d(u, v)/d_G(u, v)$ over all pairs of vertices $u, v \in V$, where $d(u, v)$ denotes the Euclidean distance between u and v $d_G(u, v)$ denotes the distance between u and v in G. That is, the dilation reflects how well the graph preserves the Euclidean distance between vertices and it therefore is a measure for the maximum detour when traveling between u and v in G.

The problem studied by Farshi et al. [FGG08] differs considerably from ours. Consider, for instance, the geometric path P over n vertices shown in Figure 4.3 consisting of n consecutive vertices $v_1, \dots, v_n$ such that the vertices, except v_2 are placed on a horizontal line. The distances between consecutive vertices on P is 1 and the distance between v_1 and v_3 is ε. Hence, P has dilation $2/\varepsilon$, attained by vertices v_1 and v_3, while its routing cost

$$2 \sum_{1 \le i < j \le n} (j - i) = \frac{(n-1)n(n+1)}{6}$$

does not depend on the geometric embedding of P. With one extra edge available, we would minimize the dilation of P by inserting it between v_1 and v_3, whereas the routing cost of P is minimized by adding an edge connecting $v := v_{n/5}$ to $v' := v_{4n/5}$. The proof of optimality requires lengthy calculations. We note however, that the routing cost that can be saved by a shortcut connecting v_1 and v_3 is only $n - 2$ while the routing cost that can be saved by connecting v and v' is at least $4n^3/125$, which results from the fact that there are $n/5$ vertices at both ends of the shortcut and each path from the first $n/5$ vertices to one of the last $n/5$ vertices saves roughly $4n/5$ vertices. However, not only the paths between vertices to the left of v and to the right of v' become shorter as vertices between v and v' also benefit from the shortcut edge, as Figure 4.2 indicates.

In this chapter we first consider the following problem, which we generalize in subsequent sections. We are given a path $P = (v_1, v_2, \dots, v_n)$ all of whose edges (v_i, v_{i+1}) are directed

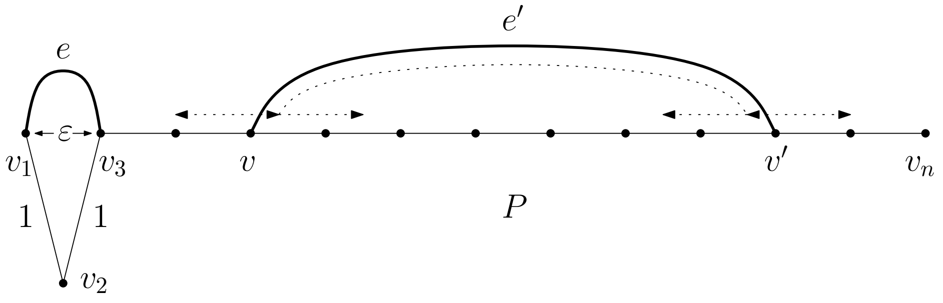


Figure 4.2: Inserting edge e minimizes the dilation, while e' minimizes the routing cost of path P.

from left to right. Each vertex v_i is assigned a weight $w_i > 0$, reflecting the number of residents of city v_i and we wish to decrease the *weighted routing cost*

$$r := \sum_{1\leq i<j\leq n} w_i(j-i)w_j$$

of P by adding one more edge (v_k, v_l), called *shortcut*, directed from v_k to v_l, choosing $k < \ell$ such that the total routing cost is minimized, that is, such that the decrease in routing cost is as large as possible as illustrated in Figure 4.3.

As compared to the undirected case shown in Figure 4.2, the situation has become simpler in that only vertex pairs on opposite sides of the shortcut edge can benefit from the extra edge. Thus, the decrease in routing cost effected by adding a directed edge from v_k to v_ℓ equals

$$\rho(k,\ell) := \left(\sum_{1\leq i\leq k} w_i\right)(\ell-k-1)\left(\sum_{\ell\leq i\leq n} w_i\right) , \tag{4.3}$$

since $\ell-k-1$ edges between v_k and v_ℓ can be avoided by using edge (v_k, v_ℓ). For unit weights, this term simplifies to $\rho(k,\ell) = k(\ell-k-1)(n-\ell+1)$. In this case, the best shortcut is a directed edge from $v_{n/3}$ to $v_{2n/3}$.

Lemma 4.1. *If $w_i = 1$ for all $1 \leq i \leq n$, then $\rho(k,\ell)$ is maximized for $k = \lfloor n/3 \rfloor$ and $\ell = n - \lfloor \frac{n}{3} \rfloor - 1$.*

Proof. In order to simplify the argumentation we set $\ell' := n-\ell+1$, that is $\ell = n-\ell'+1$ and we consider the function $\rho'(k,\ell') := \rho(k, n-\ell'+1) = k(n-\ell'-k)\ell'$. If the sum $k+\ell'$ is fixed, then the term $k(n-\ell'-k)\ell'$ only depends on the product $k\ell'$ since $n-\ell'-k$ is fixed if $k+\ell'$ is fixed. The product $k\ell'$ is maximized for $|k-\ell'| \leq 1$, since k and ℓ are positive integers. Since $\rho'(k,l')$ is symmetric with respect to k and ℓ' the maximum of ρ' is attained either for $\ell' = k$ or $\ell' = k+1$.

Simple calculations show that $\rho'(k,k) - \rho'(k,k+1) = k(2+3k-n) \geq 0$ if and only if $k \geq \frac{n-2}{3}$, since k is positive. That is, for these values of k, the maximum of $\rho'(k,\ell')$ is attained for $\ell' = k$. Assuming $\ell' = k$ we have $\rho'(k,\ell') = k^2(n-2k)$ subject to $k \leq n$, which is maximized for either $k = \lfloor \frac{n}{3} \rfloor$ or $k = \lceil \frac{n}{3} \rceil$ in the range $[1,\ldots,n]$. Assuming that $n = 3r+i$ for $r = \lfloor \frac{n}{3} \rfloor$ and $i = n-3r$, we obtain $\rho'(3r,3r) - \rho'(4r,4r) = r^2(7+74r-7n)$, which is

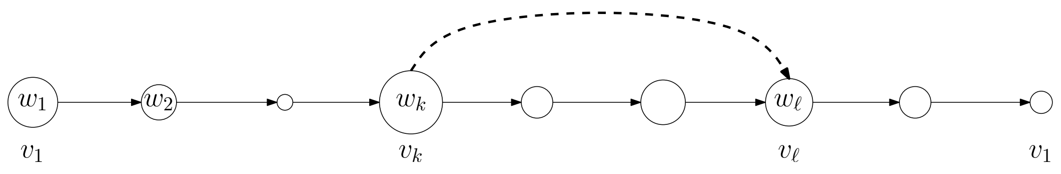


Figure 4.3: Directed, vertex-weighted path with weights w_i representing the population size of city v_i.

larger than zero, whenever $n \geq 3$. Hence $\rho'(k,k)$ is maximized for $k = \lfloor \frac{n}{3} \rfloor$. Since we have $\lfloor \frac{n}{3} \rfloor \geq \frac{n-2}{3}$, we can conclude that $\rho'(k,\ell')$ is maximized for $k = \ell' = \lfloor \frac{n}{3} \rfloor$, that is, $\rho(k,l)$ is maximized for $k = \lfloor \frac{n}{3} \rfloor$ and $\ell = n - \lfloor \frac{n}{3} \rfloor - 1$. □

Introducing non-unit weights, on the other hand, makes the maximization problem non-trivial. In fact, the decrease function $\rho(k,\ell)$ can have multiple local maxima as illustrated, for instance, by a path with 13 vertices whose weights are given by the vector $(90, 10, 100, 1, 100, 1, 1, 1, 100, 1, 100, 10, 90)$. For this path, both the shortcuts (v_3, v_9) and (v_5, v_{11}) are locally optimal.

Quite obviously, the optimum shortcut can in general be computed by inspecting the $\mathcal{O}(n^2)$ many candidates (v_k, v_ℓ) where $1 \leq k < \ell \leq n$. In this chapter, we show that we can improve on this by reducing the problem to computing the upper envelope of an arrangement of pseudo-lines in the plane if the network is a directed, weighted path. Further, we show how to extend the algorithm to directed star-shaped networks and to directed tree networks, respectively. While we obtain optimal linear-time algorithms for both directed, weighted paths and directed, weighted star-shaped networks, we only achieve a running time of $\mathcal{O}(n \log n)$ for directed trees.

4.3.1 Reduction to Pseudo-Line Arrangements

In this section we show how to compute the best shortcut for a directed, weighted path by reducing the problem to computing the upper envelope of an arrangement of piecewise linear functions. Let P denote a directed path of n vertices $v_1, \ldots, v_n$ with positive weights $w_1, \ldots, w_n$. Considering Equation 4.3 we define

$$A_k := \sum_{i=1}^{k} w_i \text{ and } B_\ell := \sum_{i=\ell}^{n} w_i$$

for $k, \ell = 1, \ldots, n$. Moreover, we let

$$f_\ell(k) := A_{\lfloor k \rfloor}(\ell - k - 1)B_\ell$$

be a function of a real variable k. In our optimization problem, only values $k \leq \ell$ are meaningful, but there is no harm in ignoring this constraint. The function $f_\ell(k)$ is piecewise linear, with discontinuities at integer values of k. For integral values of k, we have $f_\ell(k) = \rho(k,\ell)$.

A set of unbounded curves in the plane is called a family of *pseudo-lines* if each pair of curves intersect in at most one point and if the curves cross each other in this point. The following lemma proves that the family of functions f_ℓ induces a family of pseudo-lines in the plane.

Lemma 4.2. *Let $1 \leq \ell_1 < \ell_2 \leq n$ be fixed. Then the following holds.*

(i) There exists at most one real value $k \in [1, n]$ such that $f_{\ell_1}(k) = f_{\ell_2}(k)$ holds.

(ii) If $f_{\ell_1}(k) = f_{\ell_2}(k)$, then $f_{\ell_1}(k') < f_{\ell_2}(k')$ for all $k' > k$.

(iii) If no such value exists, we have $f_{\ell_1}(k') < f_{\ell_2}(k')$ for all $k' \in [1, n]$.

Proof. Consider the function

$$\begin{aligned} f_{\ell_2}(k) - f_{\ell_1}(k) &= A_{\lfloor k \rfloor}(\ell_2 - k - 1)B_{\ell_2} - A_{\lfloor k \rfloor}(\ell_1 - k - 1)B_{\ell_1} \\ &= A_{\lfloor k \rfloor}\left((\ell_2 - 1)B_{\ell_2} - (\ell_1 - 1)B_{\ell_1} + k(B_{\ell_1} - B_{\ell_2})\right) \\ &= A_{\lfloor k \rfloor}\left(C + k(B_{\ell_1} - B_{\ell_2})\right) \ , \end{aligned} \tag{4.4}$$

where C is a constant depending only on ℓ_1 and ℓ_2. Since $\ell_1 < \ell_2$ we have $B_{\ell_1} > B_{\ell_2}$. Moreover, $A_{\lfloor k \rfloor} \leq A_{\lfloor k' \rfloor}$ holds whenever $k < k'$. Thus, Equation (4.4) is strictly monotonically increasing in k, which proves Claim *(i)* and *(ii)*. If Equation (4.4) never attains the value 0, then the order relation between $f_{\ell_1}(k)$ and $f_{\ell_2}(k)$ is the same all over $[1, n]$. By setting $k := \ell_2 - 1 \in [1, n]$ and since $\ell_1 - \ell_2 < 0$ by assumption, we obtain

$$f_{\ell_1}(k) = A_{\lfloor k \rfloor}(\ell_1 - \ell_2)B_{\ell_1} < 0 = f_{\ell_2}(k) \ ,$$

which completes the proof of Claim *(iii)*. □

By Lemma 4.2, the set of graphs of the functions $f_\ell(k)$ with $1 \leq \ell \leq n$ is family of pseudo-lines over the interval $[1, n]$ since any two of them have at most one point of intersection, just as proper lines would. Note that Lemma 4.2 also yields an efficient constant-time algorithm for computing the intersection of two functions f_{ℓ_2} and f_{ℓ_1}. These functions intersect exactly if there is a value k such that function $f_{\ell_2}(k) - f_{\ell_1}(k)$ attains the value zero. However, due to Equation (4.4) this function is equivalent to

$$A_{\lfloor k \rfloor}(C + kC') = 0$$

where C and C' are constants depending only on ℓ_1 and ℓ_2. Since $A_{\lfloor k \rfloor} > 0$, this is equivalent to

$$k = -\frac{C}{C'} \ ,$$

which can be computed efficiently. Note that we are only interested in intersections in the range $[1, n]$, however.

Arrangements of pseudo-lines have been extensively studied and are reviewed in the *Handbook of Combinatorial and Computational Geometry* by Goodman [GO97], for example. We can solve our optimization problem by constructing the upper envelope of this pseudo-line arrangement, that is, the graph $\mathcal{G}$ of the maximum function

$$f(k) := \max_{1 \leq \ell \leq n} \{f_\ell(k)\} \ . \tag{4.5}$$

Each function $f_\ell(k)$ contributes at most one segment to $\mathcal{G}$. Conversely, assume that some $f_\ell(k)$ contributed two segments to $\mathcal{G}$. Then a segment of some $f_{\ell'}(k)$, where $\ell \neq \ell'$, must occur in between. However, this implies that $f_\ell(k)$ and $f_{\ell'}(k)$ must intersect twice—in contradiction to Lemma 4.2.

This is a special, and in fact the most simple, case of a Davenport-Schinzel sequence. In general, if any two of n function graphs over some interval intersect at most s times, their envelope is of complexity $\mathcal{O}(\lambda_s(n))$, with a non-trivial, slightly super-linear function λ_s as illustrated in the monograph by Sharir and Agarwal [SA95]. There is a simple algorithm that allows the lower (or upper) envelope to be constructed in time $\mathcal{O}(\lambda_s(n)\log n)$ by divide-and-conquer. Here one assumes that elementary operations, like computing an intersection of two functions, can be carried out in constant time. In our case, this is true as argued above. Moreover, $s = 1$ and $\lambda_1(n) = n$ hold.

These facts give us a first improvement over the trivial $\mathcal{O}(n^2)$ algorithm mentioned in Section 4.3. We can construct the upper envelope $\mathcal{G}$ in time $\mathcal{O}(n\log n)$. Then we perform one pass over $\mathcal{G}$ and evaluate $f(k)$ at all integer values of k in time $\mathcal{O}(n)$. If the maximum of these values is attained within a segment of $f_\ell(k)$ in $\mathcal{G}$, then the shortcut edge from v_k to v_ℓ yields a maximum reduction in routing cost.

4.3.2 Computing the Envelope in Linear Time

To improve on the $\mathcal{O}(n\log n)$ upper time bound just mentioned, we will make use of the fact that the ordering of the functions $f_\ell(k)$ "far to the right" is known to us. Indeed, for each $\ell \le n-1$, Equation (4.4) yields

$$\begin{aligned} f_{\ell+1}(n) - f_\ell(n) &= A_{\lfloor n\rfloor}(\ell+1-n-1)B_{\ell+1} - A_{\lfloor n\rfloor}(\ell-n-1)B_\ell \\ &= A_n\left((\ell-n)\sum_{i=\ell+1}^{n} w_i - (\ell-n)\sum_{i=\ell}^{n} w_i + B_\ell\right) \\ &= A_n(B_\ell + (n-\ell)w_\ell) > 0 \end{aligned}$$

so that we obtain

$$f_1(n) < f_2(n) < \cdots f_n(n)\ .$$

Next, we prove a general result concerning the computation of the upper envelope of a set of pseudo-lines, for which the order "far to the right" is known.

Theorem 4.5. *Let $\mathcal{F} = \{f_1,\dots,f_n\}$ form an arrangement A of pseudo-lines over the interval $I = [a,b]$. If the order of values $f_i(b)$ is known and if the intersection of two pseudo-lines can be computed in constant time, then the upper envelope of A can be computed in time $\mathcal{O}(n)$.*

Proof. We assume that the order of the values $f_i(b)$ for $1 \le i \le n$ is such that $f_i(b) < f_j(b)$ whenever $i < j$, that is, f_n is part of the upper envelope of $\mathcal{F}$ at b. We proceed from right to left and process the curves f_j in order of decreasing indices. In a stack S we store the part of the upper envelope that would result if no further curves existed, as illustrated in Figure 4.4. Initially, S contains only f_n. When processing f_j, we compute the intersection w of f_j with the top element f_k of S. If w does not exist, or if it lies to the left of interval $I = [a,b]$, we ignore f_j and start processing f_{j-1}.

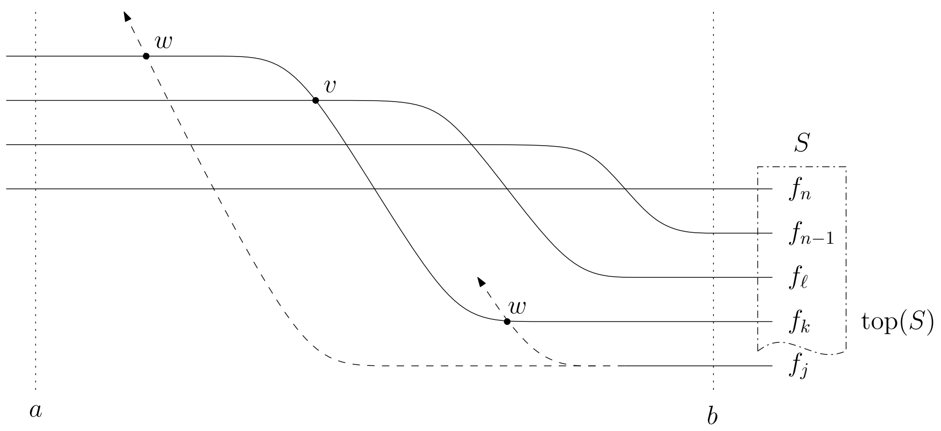


Figure 4.4: Constructing the upper envelope.

Otherwise, let v be the intersection of the two topmost curves in S. If w lies to the left of v, or if no v exists since the stack contains only one element $f_k = f_n$, then we push f_j on the stack, and start processing f_{j-1}. However, if w lies to the right of v, we pop f_k from the stack and continue processing f_j. This pop operation is justified since to the left of w, curve f_k is dominated by f_j and can, therefore, not contribute to the upper envelope. With this observation, the correctness of our algorithm is evident.

The linear running time bound can be shown as follows. During each pass through the loop described above, we (i) ignore f_j, or (ii) push f_j, or (iii) we pop f_k. Whenever (i) or (ii) occurs, we decrease the index j of the function currently processed. Thus, (i) and (ii) occur at most n times. If there are only n push operations, there can be no more than n pop operations either, because each pop is successful. Thus, the loop is carried out at most $\mathcal{O}(n)$ times, which proves the linear time bound. Upon termination, the stack S contains the segments of the upper envelope, with the leftmost segment on top. □

Using Theorem 4.5 we obtain the following result for directed weighted paths.

Theorem 4.6. *Let P be a directed, weighted path with n vertices. Then we can compute the best shortcut in terms of routing cost in $\mathcal{O}(n)$ time.*

Proof. In Section 4.3.1 we have argued that we can compute the intersection between two functions f_{ℓ_1} and f_{ℓ_2} in constant time. Thus, we can compute the upper envelope of the functions f_ℓ for $1 \leq \ell \leq n$ in linear time using Theorem 4.5. Then we can simply examine the upper envelope in linear time to compute the maximum decrease in routing cost. □

With pseudo-lines replaced by proper straight lines, our task would be to construct the upper envelope of an arrangement of lines that are sorted by slope. By duality, this task corresponds to computing the upper convex hull of a set of points sorted by x-coordinates. Under this duality, our algorithm would correspond to a variant of *Graham's* scan algorithm, namely *Andrew's monotone chain method* [And79].

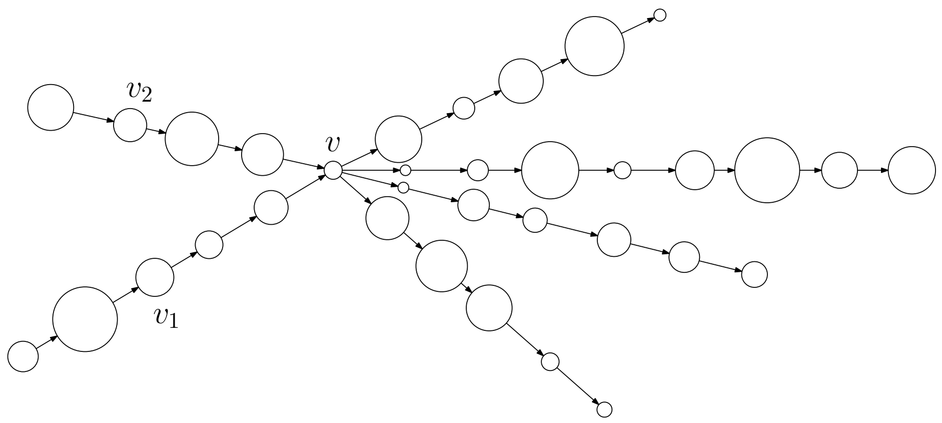


Figure 4.5: A directed star-shaped network with center v, two incoming and four outgoing paths of different length. Vertices v_1 and v_2 both have path distance 3 to center v.

4.3.3 Extension to Directed Tree Networks

Fortunately, we can apply the above optimal algorithm as a subroutine for computing the optimal shortcut for directed, weighted trees efficiently. First, we consider star-shaped trees to illustrate one of the main ideas for generalizing the algorithm presented in the previous section to more general trees. Then we show how this idea can be applied to arbitrary directed trees.

Star-shaped Networks One of the most simple extensions of a one-way path is given by a star-shaped, directed network as depicted in Figure 4.5. These networks consist of a center vertex v that has s incoming paths on m_i vertices for $i = 1, \ldots, s$ and r outgoing paths of n_j vertices for $j = 1, \ldots, r$, respectively. There are three possible locations for a shortcut. It may either be placed fully inside an incoming path, fully inside an outgoing path or it may connect an incoming path to an outgoing path. In the first two cases we can directly apply the same idea as presented above. For this to work, it suffices to additionally sum up all weights in all outgoing paths (resp., all incoming paths), and to add this sum to the weight of the branching vertex v. Then we run the linear time algorithm of Section 4.3.2 on each of the paths that result from removing v. This can be done in total time $\mathcal{O}(n)$.

The remaining task is to compute the best shortcut connecting an incoming path to an outgoing path. Obviously, one could apply the linear time algorithm for paths to all $s \times r$ path combinations, which would result in quadratic running time. The following observation helps to improve on this bound. Suppose we fix a vertex u in one of the outgoing paths to the right of v, and consider possible partner vertices to the left of v that might form a good shortcut with u. Consider two vertices v_1 and v_2 on two different incoming paths with equal distance from v as illustrated in Figure 4.5. Further, assume that the weights of the vertices in the figure are proportional to the areas of the circles. Then the sum of the weights up to v_1 is larger than that up to v_2. Choosing vertex v_1 as the left endpoint in this situation will result in a higher reduction of the routing cost for any fixed right endpoint u. That is, it suffices to know which vertex is the best left endpoint for any given distance to v.

Based on this observation, we collect the most promising vertices of the s incoming paths

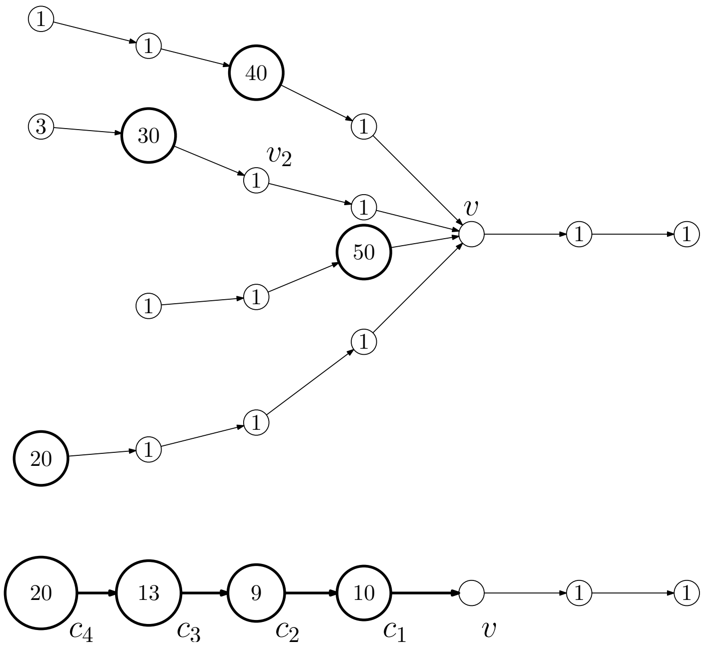


Figure 4.6: Collecting vertices into a single path. The original directed, weighted, star-shaped tree S is illustrated above. The path below is the directed, weighted, star-shaped tree S' resulting from replacing all incoming paths in S by the new collected incoming path.

into a single path, in the following way. Let $m := \max\{m_i \mid 1 \leq i \leq s\}$ denote the maximum length of an incoming path. Assume that the vertices on the i-th incoming path are labeled $v_1^i, v_2^i, \ldots, v_{m_i}^i$ from right to left, that is, with increasing distance from v. For each possible edge distance $d = 1, 2, \ldots, m$ let

$$w(d) := \max_{i=1,\ldots,s} \sum_{j=d}^{m_i} w_j^i \ , \tag{4.6}$$

where w_j^i, the weight of vertex v_j, equals zero if $j > m_i$. The new collected incoming path is defined to contain m vertices c_d with weights $w(d) - w(d+1)$ for $1 \leq d \leq m$, where $w(m+1) := 0$. An example is shown in Figure 4.6. The following lemma states some properties concerning the collected path.

Lemma 4.3. *Let S be a directed, weighted, star-shaped tree and let P be the collected incoming path of S with m vertices whose weights are defined according to Equation* (4.6)*. Then the following holds.*

(i) For each $1 \leq d \leq m$ we have $w(d) - w(d+1) > 0$.

(ii) For each $1 \leq d \leq m$, the sum of the weights of the vertices to the left of, or equal to, c_d is $w(d)$, that is, $\sum_{i=d}^{m} w(c_d) = w(d)$.

(iii) Let S' denote the directed, weighted, star-shaped tree resulting from replacing the incoming paths in S with the new collected path. Then the optimal shortcuts across v in S and S' correspond to each other.

Proof. To prove *(i)*, let i be the index of the path where $w(d+1)$ is attained. By maximality of $w(d)$, we have

$$w(d) \geq \sum_{j=d}^{m_i} w_j^i > \sum_{j=d+1}^{m_i} w_j^i = w(d+1) \ .$$

Fact (ii) follows by the telescoping property of the weights

$$\sum_{j=d}^{m} (w(i) - w(i+1)) = w(i) - w(m+1) = w(i) \ .$$

Fact *(iii)* is a consequence of *(ii)* and of our previous observation about good shortcut candidates. A shortcut (v_j^i, u) in S corresponds to a shortcut (c_i, u) in S' and a shortcut (c, d) in S' corresponds to a shortcut (v_d^i, u) such that

$$w(d) = \sum_{j=d}^{m_i} w_j^i \ .$$

$\square$

Clearly, one can also compute a collected outgoing path in the same way. Now the linear time algorithm presented in Section 4.3.2 can be run on the concatenation of the two paths, with the following modification. The functions $f_\ell(k)$ introduced in Section 4.3.1 must be considered only for indices ℓ of vertices on the collected outgoing chain, and only for arguments k corresponding to vertices on the collected incoming chain. We obtain the following result.

Theorem 4.7. *The optimum shortcut in a directed, weighted, star-shaped network can be constructed in linear time.*

General Tree Networks Based on the linear time algorithm of the previous section we now present an $\mathcal{O}(n \log n)$-time algorithm for general directed, weighted trees T of size n. We proceed by Divide and Conquer. Ignoring edge directions for the moment we find a balanced decomposition of T into two trees with exactly one vertex v in common such that each of the trees contains at least $n/3$ vertices. The existence of such a decomposition is established by the following lemma.

Lemma 4.4. *Let $T = (V, E)$ be a tree with $n \geq 2$ vertices. Then T can be decomposed into two trees T_1 and T_2 with the following properties*

(i) Both T_1 and T_2 have at least $n/3$ vertices.

(ii) There is exactly one vertex v that is contained in T_1 and T_2.

(iii) The union of T_1 and T_2 equals T.

The decomposition can be computed in $\mathcal{O}(n)$ time.

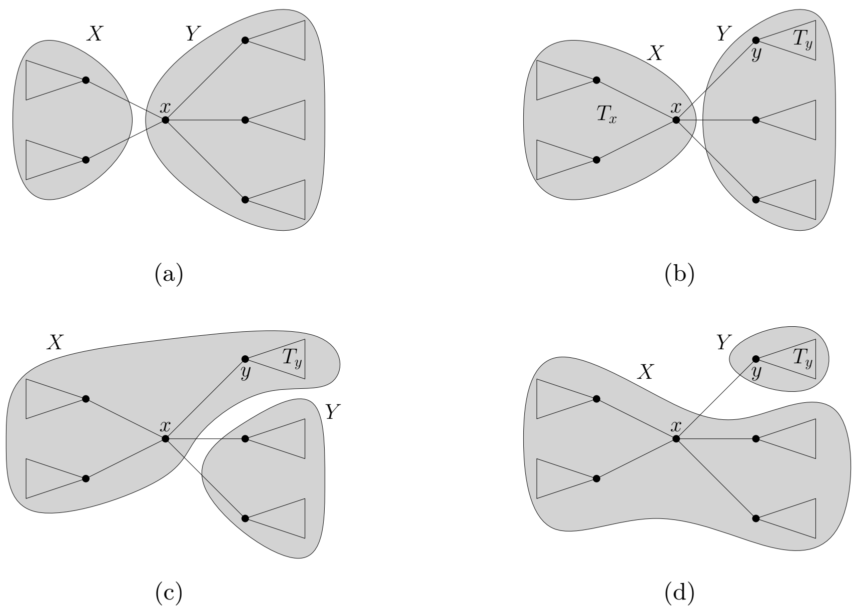


Figure 4.7: Illustration for the modification of the partition $X \uplus Y$ of the vertices of T according to the proof of Theorem 4.4.

Proof. First, we show that there is partition $X \uplus Y = V$ of the vertices of T such that each partition contains at least $n/3$ vertices and such that all edges of the tree T with one vertex in X and one vertex in Y are incident to exactly one vertex u that is either contained in X or in Y, respectively. Let uv be any edge in T and let T_u and T_v denote the trees resulting from removing uv from T. Assume without loss of generality that $|V(T_u)| \leq |V(T_v)|$. Then $X := V(T_u)$ and $Y := V(T_v)$ clearly is a partition with the desired properties. If $|X| \geq n/3$ we are done, since $|Y| \geq |X| \geq n/3$. Otherwise we show how to increase the size of X while guaranteeing that the size of Y is at least $n/3$. We are done as soon as the size of X is at least $n/3$.

So assume that $X < n/3$, that is, $Y > n/3$. First, consider the case that there is a vertex $x \in Y$ such that all edges of T between X and Y are incident to x as illustrated in Figure 4.7a. Then we remove x from Y and add it to X as illustrated in Figure 4.7b. This changes the cardinalities of X and Y by only one unit, that is, after the modification we have $|X| \leq n/3$ and $|Y| \geq n/3$. Further, all edges of T between X and Y are connected to u after the modification. If $|X| \geq n/3$ we are done.

Second, consider the case that the edges of T between X and Y are incident to at least two vertices in Y as illustrated in Figure 4.7b. Let $y \in Y$ be a vertex that is adjacent to a vertex $x \in X$. Further, let T_x and T_y be the connected trees containing x and y in $T[X]$ and $T[Y]$, respectively, and let n_x and n_y denote their respective cardinalities. If $n_y < n/3$, then $|Y| - n_y > n/3$ and we can safely move the vertices in T_y to X as illustrated in Figure 4.7c. Otherwise, $n \geq n/3$ and we can move all vertices in $Y \setminus V(T_x)$ to x as illustrated in Figure 4.7d. In both cases we increase the size of X while guaranteeing that the size of Y is at least $n/3$. Thus, the size of X will eventually reach $n/3$.

Due to the definition of the partition $X \uplus Y$ there is a vertex u such that all edges between X and Y are incident to u. Then the trees $T[X \cup \{u\}]$ and $T[Y \cup \{u\}]$ clearly satisfy the properties *(i)–(iii)* as claimed.

Finally, we show how to compute the decomposition in linear time. First, we root the tree at an arbitrary vertex v and compute for each vertex u the number of vertices n_u in the subtree rooted in u. Having these values, we can label each end of an edge e by the number of vertices in the subtrees attached to the respective ends obtained by removing e. That is, we can determine the number vertices in the respective subtrees in constant time. Note that each vertex is only moved from Y to X. Hence, we can compute the decomposition in linear time. □

In order to compute the best short cut we compute a decomposition according to Lemma 4.4 to obtain two trees T_1 and T_2 having exactly one vertex v in common. First we recursively compute the best shortcut in T_1 and T_2, respectively. Then we compute the best directed shortcut from T_1 to T_2 and vice versa. Both of them must pass through v. Let In_i be the subtree of T_i that contains all vertices from which v is reachable on a directed path for $i = 1, 2$. Similarly let Out_i be the subtrees of all vertices that can be reached from v on a directed path. We need to compute the optimum shortcut from In_1 to Out_2 and from Out_1 to In_2. These sub-tasks can be implemented as follows. Analogously to the previous section we construct the collected paths for subtrees In_i and Out_i for $i = 1, 2$. This can be done in linear time by rooting the respective trees at the common vertex and performing one top-down pass over the tree. Then we combine the corresponding collected paths into one and run the algorithm of Section 4.3.2. This yields the following result.

Theorem 4.8. *For a directed, weighted tree with n vertices, the optimal shortcut with respect to routing cost can be computed in $\mathcal{O}(n \log n)$ time.*

Proof. We have outlined the algorithm above and it only remains to show that the algorithm can be implemented to run in $\mathcal{O}(n \log n)$ time. By Lemma 4.4 we can compute the decomposition underlying the recursion in linear time. Further, the collected paths can be computed in linear time and, given the collected paths, we can compute the best shortcut in linear time as well. Next, consider the recursion tree. At each level of the tree, the total size of the sub-instances is at most $2n$ since each vertex of the tree is contained in at most two subtrees constituting the sub-instances. That is, the total amount of time needed to solve the sub-instances is $\mathcal{O}(n)$. Further, since each tree has at least $n/3$ vertices, the height of the tree is at most $\mathcal{O}(\log n)$. That is, the total running time is $\mathcal{O}(n \log n)$. □

4.3.4 Undirected Paths

Finally, we consider the undirected case of a single path. As already depicted in Figure 4.2 there are different parts of the path that might profit from the shortcut between v and v' or v_k and v_ℓ. A corresponding function $\rho(k, \ell)$ for indices $\ell < k$ becomes more complicated and our idea for the linear time algorithm is not applicable in this case. Two vertices profit, if the path length along the shortcut is smaller than the original path length. We collect the benefit of a shortcut in the following functions due to the vertices that benefit. Afterwards we will briefly explain the additional sums and the corresponding indices by an example.

First formula (from left to right as before):

$$\left(\sum_{i=1}^{k} w_i\right) \cdot \left(\sum_{j=\ell}^{n} w_j\right) \cdot (\ell - k - 1) \tag{4.7}$$

Second formula (from left to inner right):

$$\left(\sum_{i=1}^{k} w_i\right) \cdot \left(\sum_{j=\lceil \frac{\ell+k}{2}+1 \rceil}^{\ell-1} w_j \cdot (2j - \ell - k - 1)\right) \tag{4.8}$$

Third formula (from right to inner left):

$$\left(\sum_{i=\ell}^{n} w_i\right) \cdot \left(\sum_{j=k+1}^{\lfloor \frac{\ell+k}{2}-1 \rfloor} w_j \cdot (\ell + k - 2j - 1)\right) \tag{4.9}$$

Fourth formula (from inner right to inner left):

$$\left(\sum_{i=\lceil \frac{\ell+k}{2}+1 \rceil+1}^{\ell-1} w_i\right) \cdot \left(\sum_{j=k+1}^{i-\lceil \frac{k+\ell}{2}+1 \rceil+k} w_j \cdot (k - \ell + 2i - 2j - 1)\right) \tag{4.10}$$

Consider the path of Figure 4.8 with altogether 12 vertices and a shortcut from v_4 to v_{10}. Thus $\ell = 10$ and $k = 4$ holds and $\lceil \frac{\ell+k}{2} + 1 \rceil = 8$ and $\lfloor \frac{\ell+k}{2} - 1 \rfloor = 6$ is given. From inner right to the left only the vertices v_8 and v_9 profit from using the shortcut. For $j = 8, 9$ the original distance to v_4 was $j - k$ and the new distance along the shortcut is $l - j + 1$. Therefore we have a benefit of $j - k - (l - j + 1) = 2j - l - k - 1$ for the weights w_j and all weights on the left hand side ending at v_4 according to Equation (4.8).

From inner left to the right only the vertices v_5 and v_6 profit from using the shortcut. For $j = 5, 6$ the original distance to v_{10} was $l - j$ and the new distance along the shortcut is $j - k + 1$. Therefore we have a benefit of $l - j - (j - k + 1) = l + k - 2j - 1$ for the weights w_j and all weights on the right hand side starting at v_{10} according to Equation (4.9).

From inner right to inner left using the shortcut only the vertex v_9 will have a smaller distance to the vertex v_5 . We have $\lceil \frac{\ell+k}{2} + 1 \rceil + 1 = 9 = l - 1$ and $k + 1 = 5 = i - \lceil \frac{\ell+k}{2} + 1 \rceil + k$. The distance between v_i and v_j using the shortcut is given by $\ell - i + 1 + j - k$ and the direct distance is $i - j$, thus the benefit is $(i - j) - (l - i + 1 - k + j) = k - l + 2i - 2j - 1$ and Equation (4.10) provides the additional benefit in general.

Now let $\rho(k, \ell)$ be the sum of the functions of Equations (4.7)–(4.10). It can be shown that one can choose weights so that two functions $\rho(k, \ell_1)$ and $\rho(k, \ell_2)$ will have more than one intersection. This means that the key idea of Theorem 4.6 does not apply directly to undirected paths. Note, however, that the best shortcut can be easily computed in $\mathcal{O}(n^2)$ time.

4.4 Concluding Remarks

We have studied two augmentation problems for tree networks in the routing cost model. The first problem concerned a class of augmentation problems, where the goal is to find the

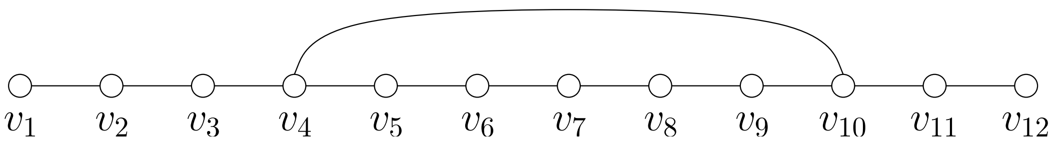


Figure 4.8: For an undirected weighted path and a shortcut different parts of the path will profit.

best connection between two disconnected trees in terms of routing cost. We presented a quadratic-time worst-case optimal algorithm for general distance functions on the vertex sets of the trees and showed that we can improve on this by providing on $\mathcal{O}(n \log n)$ time algorithm for both the Euclidean metric and sparse graph metrics

It remains an open question, for which graph metrics the problem can be solved in sub-quadratic time. Also, there are some interesting variants of the problem, for instance, when there are more than two disconnected trees. This problem arises, when a vertex of the network fails to work. Additionally, we could consider a Steiner-variant of the problem, in which we are allowed to introduce an additional vertex to which the disconnected components must be connected.

The second problem we studied was to find an optimum shortcut for oriented paths and trees with weighted vertices. We presented optimal linear-time algorithms for directed paths and star-shaped tree networks in the presence of weights and we presented an $\mathcal{O}(n \log n)$ algorithm for arbitrary directed, weighted tree networks. It is a major open problem whether this problem can be solved in sub-quadratic time on undirected tree-networks.

While both considered problems can be solved in quadratic time by exhaustive search, we were able to devise more efficient algorithms by using techniques from geometry. While we solved the augmentation problem of connecting two trees by computing a geometric and a graph-theoretic Voronoi-diagram, respectively, we solved the augmentation problem of finding the best shortcut by computing the upper envelope of an arrangement of polylines in the plane. Surprisingly, concepts from geometry helped devising efficient algorithms even for problems which have no obvious connection to geometry. We will present another example of this situation in the following Chapter 5.

Finally, we note that any generalization of these problems to more general classes of graphs than trees are likely to involve higher computational efforts. Throughout this chapter we explicitly used the fact that we can aggregate the routing cost of subtrees efficiently and subsequently work on the aggregated values instead of having to compute the routing cost effected by a single edge from scratch. However, the computation of a graph's routing cost alone involves a the computation of the shortest paths between all pairs of vertices, which has a trivial lower bound of $\Omega(n^2)$ and a non-trivial lower bound of $\Omega(nm)$ provided by Karger et al. [KKP93] for all path-comparison based all-pairs-shortest-path algorithms, that is, algorithms that are allowed to use the edge weights only for comparisons between paths.

Chapter 5

The Density Maximization Problem in Graphs

Many realistic network construction problems must find a balance between various optimization goals. Typical goals include the minimization of construction costs and the maximization of the performance of the network. We consider a framework for bi-objective network construction problems where one objective is to be maximized while the other is to be minimized. Given a *host* graph $G = (V, E)$ with edge weights $w_e \in \mathbb{Z}$ and edge lengths $\ell_e \in \mathbb{N}$ for each edge $e \in E$ we define the density of a subgraph $H = (V', E') \subseteq G$, called the *pattern* as the ratio

$$\operatorname{dens}(H) = \sum_{e \in E'} w_e / \sum_{e \in E'} \ell_e \ .$$

We consider the problem of computing a maximum density pattern H under various additional constraints. In doing so, we compute a single Pareto-optimal solution with the best weight per cost ratio subject to additional constraints further narrowing down feasible solutions for the underlying bi-objective network construction problem.

First, we consider the problem of computing a maximum density pattern with weight at least W and length at most L in a host G. We call this problem the *bi-constrained density maximization problem*. This problem can be interpreted in terms of maximizing the return on investment for network construction problems in the presence of a limited budget and a target profit. We consider this problem for different classes of hosts and patterns. Initially, we show that it is $\mathcal{NP}$-hard, even if the host has treewidth 2 and the pattern is a path. However, it can be solved in pseudo-polynomial linear time if the host has bounded treewidth and the pattern is a graph from a given minor-closed family of graphs. Finally, we present an FPTAS for a relaxation of the density maximization problem, in which we are allowed to violate the upper bound on the length at the cost of some penalty.

Second, we consider the maximum density subgraph problem under structural constraints on the vertex set that is used by the patterns. While a maximum density perfect matching can be computed efficiently in general graphs, the maximum density Steiner-subgraph problem, which requires a subset of the vertices in any feasible solution, is NP-hard and unlikely to admit a constant-factor approximation with polynomial running time. When parameterized by the number of vertices of the pattern, this problem is W[1]-hard in general graphs. On the other hand, it is FPT on planar graphs if there is no constraint on the pattern and on general graphs if the pattern is a path. This chapter is based on joint work with Mong-Jen Kao, Bastian Katz, Der-Tsai Lee, Ignaz Rutter and Dorothea Wagner [KKK^{+}11b].

5.1 Introduction

Many realistic network construction problems are characterized by complex constraints and multiple, possibly conflicting objectives, and are therefore formulated within the framework of multi-objective optimization [CP07]. There are several ways to define optimality in the context of more than one objective, among them Pareto-optimality and aggregate optimality. Let R be a multi-objective optimization problem and let I be the set of the instances of R. Further, let $f_1, \ldots, f_k \colon I \to \mathbb{R}$ be the set of objective functions of R and assume that these functions must be maximized. Then an instance $x \in I$ is *Pareto-optimal* if there is no instance $y \in I$ such that $f_i(y) \geq f_i(x)$ for all $1 \leq i \leq k$ and $f_j(y) > f_j(x)$ for at least one index $1 \leq j \leq k$. That is, x is Pareto-optimal if we cannot improve a single objective without diminishing at least one of the other objectives. While Pareto-optimality seems to capture the classical notion of optimality best, there may be many Pareto-optimal instances for a multi-objective optimization problem. Since it is sometimes undesirable to confront users with a possibly large number of Pareto-optimal solutions, it is common to combine the objectives into a single new aggregate objective, which is then optimized as a single-criterion objective. Typical aggregate functions include weighted sum or weighted minimum and maximum, that is we try to maximize

$$F(x) := \sum_{i=1}^{k} w_i \cdot f_i(x) \quad \text{or} \quad G(x) := \max\{w_i \cdot f_i(x) \mid 1 \leq i \leq k\} \; ,$$

where $w_i \in \mathbb{R}$ are constant weights. These weighted aggregate functions, however, must be guided in that the decision maker has to supply a set of suitable weights at the risk of arbitrariness.

We consider a framework for bi-objective network construction problems, motivated from economics, where one objective must be maximized while the other must be minimized. We can think of these objectives as profit and cost of an investment, respectively. Additionally, we assume that we are given an upper bound on the objective to be minimized and a lower bound on the objective to be maximized. That is, we are given a limited budget and a target profit. Two optimization functions of this sort can be aggregated by the ratio of the two optimization goals featuring two main advantages over other aggregate functions. First, we do not need to supply any weights—if we did, it would not alter our notion of optimality. Second, any optimal solution with respect to the ratio is Pareto-optimal. In economics, this ratio, termed return on investment, is a common measure for assessing the quality of investments.

Our framework is defined as follows. Let $G = (V, E)$ be a graph, which we will refer to as the *host*. Throughout the chapter we write $n := |V(G)|$ and $m := |E(G)|$, respectively. We assume that we are given a weight function $\mathrm{wt} : E \to \mathbb{Z}$ and a length function $\mathrm{len} : E \to \mathbb{N}$ on the edges, respectively. As a shorthand we write $w_e := \mathrm{wt}(e)$ and $\ell_e := \mathrm{len}(e)$ and for a subgraph $H \subseteq G$ we define

$$\mathrm{wt}(H) := \sum_{e \in E(H)} w_e \quad \text{and} \quad \mathrm{len}(H) := \sum_{e \in E(H)} \ell_e$$

as the weight and length of the edge set of a subgraph H, respectively. We refer to a subgraph of G as a *pattern*. Given $W \in \mathbb{Z}$ and $L \in \mathbb{N}$, a pattern H is called *W-viable* if $\mathrm{wt}(H) \geq W$ and it is called *(W, L)-viable* if it is W-viable and $\mathrm{len}(H) \leq L$. Our goal is to find a (W, L)-viable pattern H maximizing $\operatorname{wt}(H)$ and minimizing $\operatorname{len}(H)$, which we formalize by maximizing the ratio $\operatorname{dens}(H) = \operatorname{wt}(H)/\operatorname{len}(H)$, called the *density of H*. Given a tuple $(G, \operatorname{wt}, \operatorname{len}, W, L)$, the Bi-constrained Maximum Density Subgraph (BMDS) problem asks for a *connected* (W, L)-viable pattern $H \subseteq G$ with maximum density.

Problem Bi-constrained Maximum Density Subgraph (BMDS)

Instance: A graph $G = (V, E)$, weight $\operatorname{wt}\colon E \to \mathbb{Z}$, length $\operatorname{len}\colon E \to \mathbb{N}$, $W \in \mathbb{Z}$, $L \in \mathbb{N}$

Solution: A connected (W, L)-viable subgraph $H \subseteq G$

Goal: Maximize $\operatorname{dens}(H)$

In realistic applications, it may be desirable to be able to violate the hard limitations of our framework. For instance, it may be possible to exceed the budget by loaning additional money at the cost of some interest. We model this by introducing the *L-deviation* of H, defined as $\operatorname{dev}(H) := \max\{0, \operatorname{len}(H) - L\}$, and the *penalized density*, defined as $\operatorname{pdens}(H) := \operatorname{wt}(H)/(\operatorname{len}(H) + c \cdot \operatorname{dev}(H))$, where c is some non-negative constant. Given a tuple $(G, \operatorname{wt}, \operatorname{len}, W, L)$ the Relaxed Maximum Density Subgraph (RMDS) problem asks for a *connected* W-viable pattern $H \subseteq G$ with maximum *penalized* density. We will consider these problems for different classes of hosts and patterns.

Problem Relaxed Maximum Density Subgraph (RMDS)

Instance: A graph $G = (V, E)$, weight $\operatorname{wt}\colon E \to \mathbb{Z}$, $W \in \mathbb{Z}$, $L \in \mathbb{N}$

Solution: A connected W-viable subgraph $H \subseteq G$

Goal: Maximize $\operatorname{pdens}(H)$

A different set of constraints we consider arises from the fact that we may have to include a given set $S \subseteq V$ of vertices, so-called *terminals*, in any feasible solution, for instance, if we would like to augment an already existing infrastructure network. We will refer to these constraints as *Steiner constraints*. In the presence of Steiner constraints we drop the constraints on the length and weight of the solution.

Problem Maximum Density Steiner Subgraph

Instance: A graph $G = (V, E)$, weight $\operatorname{wt}\colon E \to \mathbb{Z}$, $W \in \mathbb{Z}$, $S \subseteq V$

Solution: A connected W-viable subgraph $H \subseteq G$ such that $S \subseteq V(H)$

Goal: Maximize $\operatorname{dens}(H)$

Related Work An overview of recent developments in multi-objective optimization is given in [CP07]. Bálint [Bál03] proves inapproximability for bi-objective network optimization problems, where the task is to minimize the diameter of a spanning subgraph with respect to a given length on the edges, subject to a limited budget on the total cost of the edges. Marathe et al. [MRS+98] study bi-objective network design problems with two minimization objectives. Given a limited budget on the first, they provide a PTAS for minimizing the second objective among a set of feasible graphs. The considered objectives include total edge weight, diameter and maximum degree.

The study of dense segments in bi-weighted sequences arises from the investigation of non-uniformity of nucleotide composition with genomic sequences [Inm66, MTB76] and has received considerable attention in bio-informatics. For this problem, we are given a sequence of pairs (a_i, b_i) and we wish to find a subsequence I with length bounded by $A \leq \sum_{i \in I} b_i \leq B$ that maximizes the density $\sum_{i \in I} a_i / \sum_{i \in I} b_i$. For uniform lengths, Lin et al. [LJC02] give an $\mathcal{O}(n \log A)$ algorithm, which is improved to $\mathcal{O}(n)$ by Goldwasser et al. [GKL05]. A linear time algorithm for the non-uniform case is given by Chung and Lu [CL05]. Lee et al. [LLL09] show how to select a subsequence whose density is closest to a given density δ in $\mathcal{O}(n \log^2 n)$ time. Without the upper bound on the length B they present an optimal $\mathcal{O}(n \log n)$-time algorithm.

Subsequently, this problem has been generalized to graphs. Previous work on this problem focuses mostly on the cases where the host is a tree subject to the two-sided constraint on the length of the solution. Hsieh et al. [HC05, HC08] show that a maximum density path in a tree subject to lower and upper length bounds can be computed in time $\mathcal{O}(Bn)$ and that it is $\mathcal{NP}$-hard to find a maximum density subtree in a tree, for which they also presented an $\mathcal{O}(B^2 n)$ time algorithm. Wu et al. [WCT99, Wu09] improve on this by presenting an optimal algorithm for computing a maximum density path in a tree in time $\mathcal{O}(n \log n)$ in the presence of both a lower and upper length bounds. They also give an $\mathcal{O}(n \log^2 n)$ algorithm for finding a *heaviest path* in a tree in the presence of length constraints [WCT99], which is improved to $\mathcal{O}(n \log n)$ by Liu and Chao [LC08] .

Problems involving Steiner constraints have been widely studied in computer science for a long time. For instance, it is known that the Steiner tree problem is NP-hard [GJ79] and can be approximated within a factor of 1.55 [RZ00]. When parameterized by the number of terminals, this problem is FPT [DW71], when parameterized by the number of non-terminals in the solution it is W[2]-hard. The latter result is attributed to Bodlaender and can be found in [Lok09]. For the special case that the set of terminals contains all vertices of the graph, Chandrasekaran [Cha77] shows that a spanning tree with maximum density can be computed in polynomial time.

Contribution In Section 5.2 we prove that the BMDS problem is $\mathcal{NP}$-hard, even if the host has treewidth 2 and the pattern is a path. Then we show how to compute a maximum density path in a tree in Section 5.2.1 and extend this result to graphs that can be turned into a tree by removing k edges, thus, showing that the problem is FPT with respect to k. In Section 5.2.2 we show how to solve the BMDS and the RMDS problems in pseudo-polynomial linear time if the host has bounded treewidth and the pattern must be contained in a given minor-closed family of graphs. Additionally, we present a general FPTAS that can be applied to all RMDS problems that admit algorithms whose running time is pseudo-polynomial in the length in Section 5.3. We show that it can be used to approximate the maximum penalized density if the host has bounded tree-width and the pattern belongs to a minor-closed family of graphs. In Section 5.4 we drop the bounds on the weight and length, respectively, and consider structural constraints instead. First, we consider general vertex constraints in Section 5.4.1. We adapt a generic technique previously used by Chandrasekaran [Cha77] for solving maximum density Steiner subgraph problems based on parametric search and we

Table 5.1: Summary of the results obtained in this chapter. The symbol $\star$ denotes an arbitrary graph.

Bi-constrained Maximum Density Subgraph Problem

G	H	Constr.	Results	Reference
$tw = 2$	path	bi-constr.	$\mathcal{NP}$-hard	Thm. 5.1
tree	path	bi-constr.	$\mathcal{O}(n \log^3 n)$	Thm. 5.2
tree +k edges	path	bi-constr	$\mathcal{O}(2^k k^2 n \log^2 n + n \log^3 n)$	Thm. 5.3
$tw = k$	minor-closed	bi-constr	$2^{\mathcal{O}(k^2 + k \log N + N)} \lvert\mathcal{F}\rvert Ln$	Thm. 5.4
$tw = k$	minor-closed	relaxed	$2^{\mathcal{O}(k^2 + k \log N + N)} \lvert\mathcal{F}\rvert m/\varepsilon^2 \log B$	Cor. 5.1

Maximum Density Steiner Subgraph

G	H	Constr.	Results	Reference
$\star$	matching	V	$\mathcal{O}((m + n \log n) n \log (nM))$	Cor. 5.4
tree	tree with k leaves	$\lvert S\rvert \geq 1$	$\mathcal{O}(k^2 n \log (nM))$	Thm. 5.7
$\star$	path	$\lvert S\rvert = 1$	$\mathcal{NP}$-hard, $\notin$ APX	Thm. 5.8
$\star$	$\star$, $\lvert V(H)\rvert \leq k$	$\lvert S\rvert = 1$	W[1]-hard	Thm. 5.9
planar	$\star$, $\lvert V(H)\rvert \leq k$	$\lvert S\rvert = 1$	FPT	Thm. 5.10
$\star$	path, $\lvert V(H)\rvert \leq k$	$\lvert S\rvert \geq 1$	$\mathcal{O}((2^{k-s} m + 3^{k-s}) s^2 \log (nM))$	Thm. 5.11
$\star$	tree	$\lvert S\rvert \geq 1$	$\mathcal{NP}$-hard	Thm. 5.12

show how this technique can be used to solve the maximum density perfect matching problem efficiently. Further, we show how to compute a maximum density tree with k leaves in a tree in polynomial time. Then we focus on Steiner constraints requiring a given subset S of the vertices in any feasible solution in Section 5.4.2. We show that this problem is NP-hard and cannot be approximated by a constant factor unless $P = NP$, even if the pattern is a path and S contains only one vertex. Further, when parameterized by the number of vertices of the pattern, we show that the maximum density subgraph problem is W[1]-hard, that is, it is unlikely to admit an FPT-algorithm. In contrast, we show that this problem is FPT on planar graphs. Then we show that problem of computing a maximum density path is FPT when parameterized by the number of vertices on the path in general graphs. However, we also show that it is NP-hard to find a maximum density Steiner tree.

5.2 The Bi-constrained Maximum Density Subgraph Problem

In this section we consider the Bi-constrained Maximum Density Subgraph (BMDS) problem. Given an instance $I = (G, \mathrm{wt}, \mathrm{len}, W, L)$ we wish to find a connected (W, L)-viable subgraph of G with maximum density. Since the BMDS problem can be solved in time $\mathcal{O}(n^2)$ when the host is a tree and the pattern is a path by enumerating all possible paths, it is natural to ask if the BMDS problem can be solved efficiently on more general hosts and patterns. However, we show that it is $\mathcal{NP}$-hard to find a maximum density path, even if the

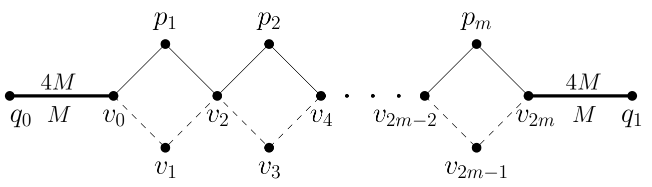


Figure 5.1: Graph used in the reduction from Partition. Bold edges have density 4, all other edges have density 1. Dashed edges have weight and length 1, solid non-bold edges incident to p_i have weight and length $c_i + 1$.

host is only slightly more complicated than a tree.

Theorem 5.1. BMDS *is $\mathcal{NP}$-hard, even if the host is a simply connected outerplanar graph with treewidth 2, the pattern is a path and we drop the upper bound on the length of the pattern.*

Proof. The proof is by reduction from Partition. Assume we are given an instance of Partition, that is, a set of positive integers $C = \{c_1, c_2, \ldots, c_m\}$ with $M = \sum_{i=1}^{m} c_i$ and we ask whether there is a subset $I' \subseteq \{1, \ldots, m\}$ with $\sum_{c_i \in I'} c_i = M/2$. We transform this into an instance of BMDS as illustrated in Figure 5.1. First, we create a path $v_0, v_1, \ldots, v_{2m}$ with $w_e = \ell_e = 1$ for each edge e on this path. Besides, we create additional m vertices, $p_1, p_2, \ldots, p_m$, and connect p_i to both v_{2i-2} and v_{2i} with $w_e = \ell_e = c_i + 1$ for $e \in \{p_i v_{2i-2}, p_i v_{2i}\}$. Then we create additional vertices q_0 and q_1, which we connect to v_0 and v_{2m}, respectively, such that $w_{q_0 v_0} = w_{q_1 v_{2m}} = 4M$ and $\ell_{q_0 v_0} = \ell_{q_1 v_{2m}} = M$. Furthermore, we set $W = 9M + 2m$. Since the graph is outerplanar, its treewidth is bounded by 2.

We claim that there is a path with length at least W and density at least $d := W/(3M+2m)$ if and only if the corresponding instance of Partition can be solved. Clearly, any partition can be transformed into a path with density d and length W. Let (C_1, C_2) be a partition of C and let S denote the indices of the elements in C_1, that is, $\sum_{i \in S} c_i = M/2$. Consider the simple path P from q_0 to q_1 that visits all vertices v_i with $i \in S$ and that contains none of the vertices v_j with $j \notin S$. This path has weight $8M + 2m + 2\sum_{i \in S} c_i = 9M + 2m$ and length $2M + 2m + \sum_{i \in S} c_i = 3M + 2m$. Hence, it has density d.

Conversely, assume that P is a path with density at least d. Since the weight of the path must be at least $9M$ it must end at q_0 and q_1, respectively. Let S be the set of indices such that p_i is on the path if and only if $i \in S$. Then the density of this path can be expressed as

$$\frac{8M + 2m + 2\sum_{i \in S} c_i}{2M + 2m + 2\sum_{i \in S} c_i},$$

which is strictly decreasing as $2\sum_{i \in S} c_i$ is increasing. Hence we have $2\sum_{i \in S} c_i \leq M$. On the other hand the weight of the path must be at least W, which implies $2\sum_{i \in S} c_i \geq M$. Thus, the path induces a valid partition $I' := \{c_i \mid i \in S\}$. □

When both the lower-bounded and upper-bounded constraints are imposed the problem becomes much harder. By setting $L = 3M + 2m$ in the reduction above we can show that it is $\mathcal{NP}$-hard to even compute any feasible solution if we impose both the lower bound on the weight and the upper bound on the length of the pattern. Hence, the problem is not likely to be approximable in polynomial time.

5.2.1 Density Maximization for Trees and Almost-Trees

In the previous section, we have shown that it is $\mathcal{NP}$-hard to compute a maximum density path even if the host graph has treewidth 2. Hence, the problem is unlikely to be FPT with respect to the parameter treewidth. In this section, however, we show that the problem of computing a maximum density path is FPT with respect to the number of edges k that must be deleted from a graph in order to obtain a tree. The treewidth of such a graph is bounded by $k+1$. We prove this result in two steps. First, we show how to compute a maximum density path when the host is a tree. The problem can trivially be solved in $\mathcal{O}(n^2)$ time by enumerating all possible paths, but we show how to solve it in $\mathcal{O}(n \log^3 n)$ time. Our basic approach is similar to one described by Wu [Wu09] and Lau et al. [LNN06] with respect to decomposing the problem into smaller sub-problems. However, we use completely different techniques for the sub-problems to obtain our results, since the results by Wu and Lau et al. are not applicable in our setting. Second, we use this result to show that finding a maximum density path in a general graph is FPT with respect to the number of edges we have to remove in order to obtain a tree.

Throughout the section we use the key idea that the combined density of two subpaths P and Q is equal to the slope between two points $u_P = (\operatorname{len}(P), \operatorname{wt}(P))$ and $-u_Q = (-\operatorname{len}(Q), -\operatorname{wt}(Q))$ in the Euclidean plane. This path is feasible if and only if $\operatorname{wt}(P) \geq W - \operatorname{wt}(Q)$ and $\operatorname{len}(P) \leq L - \operatorname{len}(Q)$. For a given query path Q this slope is maximized on the convex hull of the set of points $\mathcal{P}_Q$ representing the candidate subpaths for Q in the range $(-\infty, L - \operatorname{len}(Q)] \times [W - \operatorname{wt}(Q), \infty)$ as illustrated in Figure 5.2a. Since we wish to maximize the density it suffices to perform tangent queries to the upper chain of the convex hull of $\mathcal{P}_Q$, denoted by $\operatorname{UH}(\mathcal{P}_Q)$, which is more efficient than trying all possible combinations.

We use a dynamic data structure for the maintenance of the upper chain of the convex hull of a set of points P [OvL81] allowing point insertions in time $\mathcal{O}(\log^2 n)$. It maintains the upper chain of the convex hull by a dynamically maintained ordered binary tree. Each leaf of this tree corresponds to a point in the plane and each inner node v corresponds to the segment of the upper hull of the set of points P_v that does not contribute to the upper hull of the set of points in the subtree of the father of v. Each inner node additionally stores the number of points on its upper hull that it inherits from its father and, whether these points are at its left or right boundary. The segments of the upper hulls are represented by concatenable queues which allow insertion, deletion, concatenation and split in $\mathcal{O}(\log n)$ time. Lemma 5.1 shows how to compute $\operatorname{UH}(\mathcal{P}_Q)$ from a given set of candidate paths $\mathcal{P}$ in time $\mathcal{O}(\log^2 n)$ given the dynamic data structure.

Lemma 5.1. *Given a point Q, a set of points $\mathcal{P}$ and a dynamic data structure for the maintenance of $\operatorname{UH}(\mathcal{P})$ as described in [OvL81] the upper convex hull $\operatorname{UH}(\mathcal{P}_Q)$ can be computed in time $\mathcal{O}(\log^2 n)$.*

Proof. If $\mathcal{P}_Q$ is empty, there is nothing to do. Otherwise, let $p_{\min}$ be the leftmost point in $\mathcal{P}_Q$ and let $p_{\max}$ be the rightmost point in $\mathcal{P}_Q$. Let $x_{\min}$ and $x_{\max}$ be the respective x-coordinates of these points. Let $\mathcal{P}' := \{(x, y) \in \mathcal{P} \mid x_{\min} \leq x \leq x_{\max}\}$. First, we prove that $\operatorname{UH}(\mathcal{P}_Q) \subseteq \operatorname{UH}(\mathcal{P}')$. Clearly, $\mathcal{P}' = \mathcal{P}_Q \uplus \mathcal{P}''$ where $\mathcal{P}''$ contains all the points $(x, y) \in \mathcal{P}$ with $x_{\min} \leq x \leq x_{\max}$ and $y < W$. Thus, all points in $\mathcal{P}''$ are either in the interior of the convex hull of $\mathcal{P}_Q$ or on a vertical line through $x_{\min}$ or $x_{\max}$, respectively. Hence, the claim holds and we can reduce the problem of computing $\operatorname{UH}(\mathcal{P}_Q)$ to computing the upper hull of

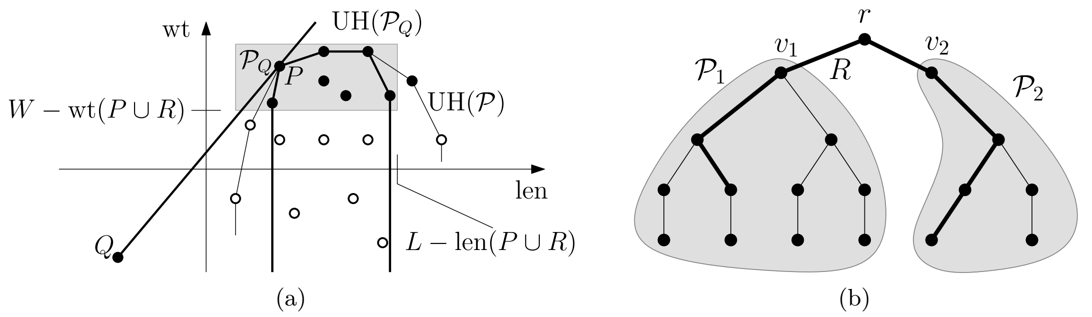


Figure 5.2: Tangent query to find the best candidate for Q (a) and combination of two paths using vertex r (b).

a set of points $\mathcal{P}'$ in a vertical strip of the plane, which is supported by the dynamic data structure. Let $\mathcal{T}$ denote the tree used by the dynamic data structure for the maintenance of the upper hull. In order to compute UH($\mathcal{P}'$) we traverse the paths from the root of $\mathcal{T}$ to $p_{\min}$ and $p_{\max}$, respectively, in parallel. In each step we reconstruct the upper hull using the concatenation and split operation of the concatenable queues stored in the nodes of the tree. We split off branches of the tree that are to the left of the path from the root to $p_{\min}$ and to the right of the path from the root to $p_{\max}$. These branches contain only points whose x-coordinates are either greater than $x_{\max}$ or smaller than $x_{\min}$. Since $\mathcal{T}$ is balanced we have reconstructed UH($\mathcal{P}'$) after at most $\log n$ steps using time $\mathcal{O}(\log n)$ per step and $\mathcal{O}(\log^2 n)$ time in total. Clearly, we can reconstruct the original data structure with the same complexity. □

Theorem 5.2. *Given an instance* $(T, \mathrm{wt}, \mathrm{len}, W, L)$ *of the* BMDS *problem, where* $T = (V, E)$ *is a tree, we can compute a* (W, L)*-viable maximum density path in* $\mathcal{O}(n \log^3 n)$ *time.*

Proof. Without loss of generality we may assume that T is a binary tree. Otherwise we can make it binary by adding auxiliary vertices and edges with weight and length 0 in linear time such that the resulting tree has linear size. A *centroid* of a binary tree is a vertex whose removal disconnects T into at most three subtrees with at most half of the vertices of the original tree in each of the subtrees. We root T in one of its centroids r. Clearly, a centroid can be computed in linear time by aggregating weights of the tree starting in the leaves. Let v_1, v_2 be two children of r and let R be the path between v_1 and v_2 via r as illustrated in Figure 5.2b. Then we can compute the maximum density path including R using tangent queries in time $\mathcal{O}(n \log^2 n)$ as follows. First, we compute the set $\mathcal{P}_1$ of paths starting in v_1 and compute their density in linear time. Each of those paths $P \in \mathcal{P}_1$ is mapped to a point $u_P := (\mathrm{len}(P \cup R), \mathrm{wt}(P \cup R))$ in the plane and inserted into the dynamic datastructure for the maintenance of the upper hull. This can be done in $\mathcal{O}(n \log^2 n)$. Then we compute the set of paths $\mathcal{P}_2$ starting in v_2. For each of these paths $Q \in \mathcal{P}_2$ we want to compute the best path $P \in \mathcal{P}_1$, that is, a path P such that the concatenation of Q and $P \cup R$ has maximum density.

To this end, we map each $Q \in \mathcal{P}_2$ to a point $-u_Q := (-\mathrm{len}(Q), -\mathrm{wt}(Q))$. Since we have bounds on both the weight and the length of a feasible solution, not all paths in $\mathcal{P}_1$ will be feasible partners for a given $Q \in \mathcal{P}_2$. We require that the length of $P \in \mathcal{P}_1$ is bounded by $\mathrm{wt}(P) \geq W - \mathrm{wt}(Q) - \mathrm{wt}(R)$ and $\mathrm{len}(P) \leq L - \mathrm{len}(Q) - \mathrm{len}(R)$. Using Lemma 5.1 we can compute the maximum density partner for $Q \in \mathcal{P}_2$ in time $\mathcal{O}(\log^2 n)$. During of the computation of the upper convex hull we can simultaneously perform the necessary tangent queries using binary search on the constructed hull. Then we can compute the maximum density path P^* through r in time $\mathcal{O}(n \log^2 n)$. We do this for all (at most 6) combinations of children of r and store the path of maximum density. Next, we recursively compute the best path through each of the children of r in the subtrees rooted in the children. Let $\hat{P}$ be the maximum density path over all the paths computed this way. Then the maximum density path in the tree rooted in r is the maximum density path over P^* and $\hat{P}$. The recurrence relation for the computation is given by $T(n) = \sum_{i=1}^{3} T(n_i) + \mathcal{O}(n \log^2 n)$, where n_i is the number of vertices in the tree rooted in v_i. Hence, the running time of this approach is $\mathcal{O}(n \log^3 n)$. ☐

Next, we show that we can obtain a similar result if the host is a graph that can be turned into a tree by deleting a fixed number of k edges. Roughly, the key idea consists of enumerating all possible subsets of the k edges and computing, for each of those subsets, the maximum density path containing all these edges. The following lemma can be used to enumerate these paths efficiently.

Lemma 5.2. *Given a graph $G = (V, E \cup F)$ such that $T = (V, E)$ is a tree and $F \cap E = \emptyset$ as well as $F' \subseteq F$ and vertices $s, t \in V$ incident to the edges in F', then there is at most one s-t-path in G containing all edges in F'. We can compute such a path or conclude that no path exists in linear time.*

Proof. We prove the existence of at most one path by contradiction. Suppose that there are two different paths P_1 and P_2 both containing all edges in F' and ending with s and t, respectively. Since the paths are different and both contain all edges in F' the symmetric difference Δ of $E(P_1)$ and $E(P_2)$ is non-empty and contained in E. Since both paths end at s and t, all vertices of Δ have even degree. Hence, Δ contains a cycle contradicting the fact that Δ is a subgraph of T.

We proceed by showing that the uniquely determined feasible path can be computed in linear time, if it exists. We root T in some vertex $r \in V(T)$. By T_v we denote the tree rooted in $v \in V(T)$. For a given $F' \subseteq F$ we call $v \in V(T) \setminus \{s, t\}$ a *loose end* if it is incident to exactly one edge in F'. To compute P we traverse T in a bottom-up fashion constructing P by iteratively matching loose ends. For each vertex v we store a reference to the unmatched loose end, if it exists. Let v be a vertex with children $w_1, \ldots, w_\ell$. Clearly, there can only be a valid path if at most two children, say, w_1 and w_2, contain an unmatched loose end in their subtrees. Otherwise there is no feasible path. If none of the children contains an unmatched loose end, then there is nothing to do. If exactly one child contains an unmatched loose end in its subtree, we store a reference to this vertex in v. If exactly two children of v contain unmatched loose ends ℓ_1 and ℓ_2 in their subtrees, then we update the path by matching these loose ends and adding the unique path in T that connects ℓ_1 and ℓ_2. We accept the resulting graph if it is a path, which can be checked in linear time. ☐

Theorem 5.3. *Given an instance $(G, \mathrm{wt}, \mathrm{len}, W, L)$ of the* BMDS *problem such that G is a tree with k additional edges, we can compute a maximum density (W, L)-viable path in time $\mathcal{O}(2^k k^2 n \log^2 n + n \log^3 n)$.*

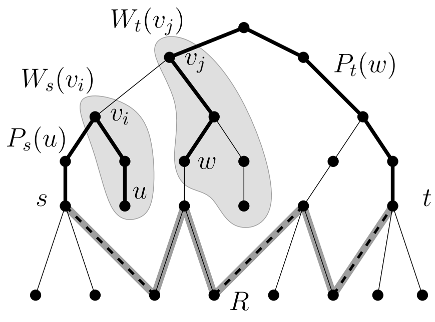


Figure 5.3: Illustration for the proof of Theorem 5.3. Extension of the unique path R using all edges in F (dashed) with end vertices s and t.

Proof. Given a tree with k additional edges $G = (V, E)$, we first compute an arbitrary spanning tree $T = (V, E')$ of G. This leaves exactly k edges, denoted by $F := E \setminus E'$, which may or may not be used by the optimal path. For each $F' \subseteq F$ we compute the maximum density path containing all edges in F' and we return the maximum density path over all $F' \subseteq F$. First, we compute the maximum density path in T in $\mathcal{O}(n \log^3 n)$ time using Theorem 5.2. Any path containing some non-empty subset of edges $F' \subseteq F$ can be decomposed into three subpaths P, Q and R such that R starts and ends with edges in F' and contains all edges in F'. By Lemma 5.2 the possible paths R are uniquely determined by choosing a set $F' \subseteq F$ as well as two vertices incident to F' and R can be computed in linear time from this information.

Hence, we iterate over all possible $F' \subseteq F$ and all $s, t \in V$ incident to F'. In each of the $2^k k^2$ iterations, we first compute both weight and length of R in linear time, resulting in $\mathcal{O}(2^k k^2 n)$ time. Then we find paths P and Q starting at s and t, respectively, such that the density of the concatenation of P, R and Q has maximum density among all paths including R in time $\mathcal{O}(\log^2 n)$. For the remainder of the proof we show how this can be accomplished and we thus assume that R, s and t are fixed. Our approach is similar to the proof of Theorem 5.2. However, we must take care of the disjointness of the paths.

Let $s = v_0, \ldots, v_\ell = t$ be the sequence of vertices on the path from s to t in T. For each of these vertices $v_i \neq s, t$, we define $W_s(v)$ as the set of vertices in T_{v_i} that are reachable from s in T without crossing the path R. Each of the vertices $w \in W_s(v_i)$ defines a path $P_s(w)$. Analogously, we define the set of vertices $W_t(v_i)$ in T_{v_i} that are reachable from t in T without crossing R. Each of those vertices $w \in W_t(v_i)$ defines a path $P_t(w)$ from t to w. Two paths in $P_s(v_i)$ and $P_t(v_j)$, respectively, are disjoint, whenever v_i is encountered before v_j on the path from s to t, that is, if $i < j$, otherwise they will have at least one vertex in common as illustrated in Figure 5.3.

Now we describe how we insert the paths into the dynamic data structure for the maintenance of the upper hull. As pointed out, paths may not be disjoint, hence, we must insert the paths in a specific order. First, we insert all paths starting in s that do not include any vertex on the path from s to t. Then, for each $i = 1, \ldots, \ell - 1$ we insert all paths $P_s(w)$ for all $w \in W_s(v_i)$. After inserting the paths for a specific $i < \ell - 2$ we make tangent queries for all paths $P_t(w)$ for $w \in W_t(v_{i+1})$. Note that at that point, we have included all paths starting in s except those that would not be disjoint to the paths in $P_s(w)$ for $w \in W_t(v_{i+1})$. After we have inserted all paths $P_s(w)$ for all $w \in W_s(v_{\ell-1})$ we have inserted all paths starting in s, which do not cross R. Then we make tangent queries for all paths starting in t that do not use any vertex on the path from s to t.

In order to compute the best path for $F = \emptyset$ we proposed an algorithm with running time $\mathcal{O}(n \log^3 n)$. For each specific non-empty choice of $F' \subseteq F$ and vertices s and t incident to F' we thus insert at most n points into the data structure with a total running time of $\mathcal{O}(n \log^2 n)$ and we perform at most n tangent queries, each with a running time of at most $\mathcal{O}(\log^2 n)$. Hence, the overall running time is $\mathcal{O}(2^k k^2 n \log^2 n + n \log^3 n)$. □

Note, that the Relaxed Maximum Density Subgraph problem can be solved within the same asymptotic bounds by similar means if the pattern is a path and the host is a tree or a tree with k additional edges, respectively.

5.2.2 Density Maximization in Graphs with Bounded Treewidth

In this section we show that a large class of problems can be solved in pseudo-polynomial FPT time when parameterized by the treewidth k of the host, that is, in time $\mathcal{O}(f(k)p(L, n))$ where f is a function depending only on k and p is a polynomial depending on the maximum length L of any feasible pattern and the number of vertices n of the graph. In the light of the results on the hardness of the problem this seems to be the best we can hope for. Given a graph G with treewidth k and a finite set of graphs $\mathcal{F}$, we wish to find a *connected* (W, L)-viable pattern H with maximum density that does not contain any graph in $\mathcal{F}$ as a minor. Such a graph is called $\mathcal{F}$-minor-free. This includes trees, (outer-)planar graphs as well as graphs from various other minor-closed families of graphs. We give an algorithm for the general case but note that the running time can be improved by considering special classes of graphs. We assume that we are given a tree decomposition of the host as an input; otherwise it can be computed in FPT time [Bod93, Klo94] with respect to the treewidth of the graph. We note that the size of the forbidden obstructions is small for many interesting examples, such as trees and (outer-)planar graphs whose sets of forbidden minors include graphs with ≤ 6 vertices.

Our algorithm is based on dynamic programming on the tree decomposition of the graph and is inspired by Eppstein's work on subgraph isomorphism in planar graphs [Epp95]. Based on Eppstein's idea of enumerating partial isomorphisms for the bags of the tree decomposition, we enumerate partial minors of the graphs induced by the bags. In the following we present the key ideas in more detail. Let $G = (V, E)$ be a graph. Recall that a *tree decomposition* of G is a pair $(\mathcal{X}, \mathcal{T})$ where $\mathcal{X} = \{X_i \mid i \in I\}$ is a collection of subsets of V which are called *bags* and $\mathcal{T} = (I, E_\mathcal{T})$ is a tree with the following properties.

(i) The union of all bags $\bigcup_{i \in I} X_i$ is equal to V.

(ii) For all edges $e \in E$ there is an index $i \in I$ such that $e \subseteq X_i$.

(iii) For all vertices $v \in V$, the tree induced by the set of nodes $\mathcal{X}_v = \{i \in I \mid v \in X_i\}$ induces a connected subtree of $\mathcal{T}$.

We will refer to the elements in I as nodes—as opposed to vertices in the original graph. The *treewidth* of a tree decomposition equals $\max_{i \in I} |X_i| - 1$. The treewidth of a graph $G = (V, E)$ is equal to the minimum treewidth of a tree decomposition of G.

Theorem 5.4. *Let $(G, \mathrm{wt}, \mathrm{len}, W, L)$ be an instance of the* BMDS *problem such that G has treewidth at most k and let $\mathcal{F}$ be a non-empty finite set of graphs. Then the maximum density connected $\mathcal{F}$-minor-free (W, L)-viable pattern can be computed in time $2^{\mathcal{O}(k^2+k\log N+N)}|\mathcal{F}|Ln$ where $N = \max_{F\in\mathcal{F}} |V(F)|$.*

Proof. We describe the algorithm for the case that $\mathcal{F}$ consists of only one forbidden obstruction F. The extension to a larger family of forbidden obstructions is straightforward. Our algorithm is by dynamic programming on the tree decomposition of the graph. Robertson and Seymour have proven the existence of an $\mathcal{O}(n^3)$ graph minor test for any fixed minor F [RS95]. However, the proof is non-constructive and involves huge constants. Therefore, we describe an explicit algorithm for the graph minor test which relies on the enumeration of subgraphs instead. We note, however, that we need some explicit representation of the minor mappings for the dynamic programming anyway, thus, this does not change the asymptotic complexity of our approach.

For the proof we assume that we are given a nice tree decomposition. A tree decomposition is called *nice* if $\mathcal{T}$ is a rooted binary tree where each node is of one of the following types: A *leaf node* X contains only one vertex. An *introduce node* X has only one child Y such that $X = Y \cup \{v\}$ for some $v \in V$. We say X *introduces* v. A *forget node* X has only one child Y such that $X = Y \setminus \{v\}$ for some $v \in V$. We say X *forgets* v. Finally, a *join node* X has two children Y_1 and Y_2 such that $X = Y_1 = Y_2$. Given a graph with treewidth k, we can always find a nice tree decomposition with $\mathcal{O}(n)$ nodes in linear time [Klo94].

Throughout the proof we assume that $G = (V, E)$ is a graph with treewidth at most k and we let $(\mathcal{X}, \mathcal{T})$ be a nice tree decomposition of G with treewidth at most k, rooted in a node $r \in I$. For $i \in I$ we denote the graph induced by the union of the bags of all descendants of i (including i) by G_i. Using standard notation, we denote the graph induced by X_i by $G[X_i]$. Let $\mathcal{C} := \{V_1, \dots, V_q\}$ be a disjoint collection of connected subsets of V. Recall that we denote the graph obtained by contracting the vertices in each of the sets V_i into a single vertex by $G/\mathcal{C}$ and that we refer to the sets V_i as *branch sets* and to $\mathcal{C}$ as a *contraction set*. Then F is a minor of G iff there is a subgraph H and a contraction set $\mathcal{C}$ such that $H/\mathcal{C}$ is isomorphic to F.

Let $\mathcal{C}$ be a contraction set and let H be some subgraph in $G[X_i]$ for some $i \in I$. A *partial minor embedding of F into H with respect to $\mathcal{C}$* is a mapping $\varphi : V(F) \to V(H/\mathcal{C}) \cup \{\bot, \top\}$ such that $uv \in E(F) \Rightarrow \varphi(u)\varphi(v) \in E(H/\mathcal{C})$ for all $uv \in E(F)$ with $u, v \notin \varphi^{-1}(\bot) \cup \varphi^{-1}(\top)$, hence, φ maps a subgraph of F to a minor of H. The image $\bot$ represents vertices in $G_i - X_i$ and the image $\top$ represents vertices in G which have not been considered yet, that is, vertices in $G - G_i$. A partial minor embedding φ is called *proper* if and only if $\varphi^{-1}(\top) \neq \emptyset$. Otherwise it represents a minor embedding of F into some subgraph of G_i, and hence, H must be disregarded as a partial solution.

For the algorithm we identify the vertices of F with the numbers $1, \dots, |V(F)|$. The images of these vertices under φ that are not contained in $\varphi^{-1}(\bot) \cup \varphi^{-1}(\top)$ correspond to branch-sets of H, that is, a partition of the vertices of H. By considering all $|V(F) + 1|^{k+1}$ labelings of the vertex set of H where each vertex is labeled with some number in $0, \dots, V(F)$, we obtain a partition of the vertices induced by the labeling. Such a partition is valid only if each set of vertices forms a connected set. Further, it defines an implicit mapping f of a subset of the vertices of F to the partitions induced by the labeling. Vertices labeled 0 are considered not to be images under f. We further encode for each vertex in F that does not have an image under f whether it is mapped to $\perp$ or $\top$. A mapping to $\perp$ means that the vertex can be mapped to some branch-set in the subgraph induced by the descendants of node i whereas a mapping to $\top$ means that we will try to map the vertex to some branch-set we have not encountered, yet. We can check in $\mathcal{O}(k^2)$ time if such an encoding represents a valid partial minor embedding of F into H. We check connectedness of the partitions in time $\mathcal{O}(k)$. Further, we check if each edge in F is represented by some edge between the corresponding branch-sets in H. This can be done in time $\mathcal{O}(k^2)$ by iterating over all pairs of vertices in H, and hence, over all pairs of labels and checking corresponding edges in both F and H. Using these conventions, we can encode a partial minor embedding φ. It is not hard to see that there are at most $|V(F)|^{k+1}2^{|V(F)|}$ many partial minor embeddings using this kind of encoding.

By $W(i, H, \Phi, \ell)$ we denote the maximum weight of a subgraph G' of G_i with length ℓ, such that $G'[X_i] = H \subseteq G[X_i]$ and Φ represents all partial minor embeddings φ of F into G'. We call the quadruple (i, H, Φ, ℓ) an *interface for i*. An interface is called *proper* if and only if $\varphi(\top) \neq \emptyset$ for all $\varphi \in \Phi$.

By G_i^* we denote the graph obtained by adding new vertices $\top$ and $\perp$ to $G[X_i]$ which are each connected to all vertices in X_i. Both weight and length of the additional edges is equal to zero. We then consider connected subgraphs in G_i^*. Note that any *connected* subgraph G' can be mapped to a *connected* subgraph in G_i^*.

The solution we are looking for will be the maximum over all interfaces (r, H, Φ, ℓ) where r is the root of the tree decomposition, such that H is connected and does not contain $\top$, Φ is proper and ℓ is at most L. If the maximum weight is at least W, then we return this weight, otherwise there is no feasible solution. We now describe how $W(i, H, \Phi, \ell)$ can be computed in $\mathcal{T}$ in a bottom-up fashion by dynamic programming starting at the leaves of $\mathcal{T}$.

Leaf node i with $X_i = \{v\}$: For each subgraph H in G_i^* that does not include $\perp$ we compute the set Φ of partial minor embeddings of F into H and we set $W(i, H, \Phi, 0) = 0$, since both the weight and length of any subgraph of G_i^* are equal to 0 by construction. Note that any vertex in F which is not mapped to v must be mapped to $\top$. Hence, the time complexity is asymptotically bounded by

$$\underbrace{\mathcal{O}(1)}_{\text{subgraphs } H} \cdot \underbrace{|V(F)|^2 \cdot |V(F)|}_{|\Phi|} \cdot \underbrace{\mathcal{O}(1)}_{\text{check mapping}} \cdot L\,.$$

Introduce node i introducing v: Let j be the only child of i and let (j, H', Φ', ℓ') be an interface for j such that $W_j := W(j, H', \Phi', \ell')$. We consider all connected subgraphs H of G_i^* which can be obtained from H' by adding v and some set of edges E^+ incident to both v and some set of vertices in H'. For each fixed H obtained this way, we further consider the set Φ of all partial minor embeddings φ of F into H that can be obtained from some $\varphi' \in \Phi'$ by choosing some vertex in $\varphi'^{-1}(\top)$ to be mapped to v by φ. If all partial minor embeddings φ constructed this way are proper and $\ell := \ell' + \mathrm{len}(E^+) \leq L$, the interface (i, H, Φ, ℓ) is proper, and we compute $W(i, H, \Phi, \ell) = W_j + \mathrm{wt}(E^+)$ and set $W(i, H', \Phi', \ell') = W_j$. Hence, the complexity for an introduce node is asymptotically bounded by

$$\underbrace{2^{\binom{k}{2}}}_{\text{subgraphs } H} \cdot \underbrace{|V(F)|^{k+1} \cdot 2^{|V(F)|}}_{|\Phi'|} \cdot \underbrace{2^k}_{|E^+|} \cdot \underbrace{|V(F)|}_{\text{new mappings to } v} \cdot \underbrace{k^2}_{\text{check mapping}} \cdot L$$

Forget node i forgetting v: Let j be the only child of i and let (j, H', Φ', ℓ') be an interface for j such that $W_j := W(j, H', \Phi', \ell')$. If H' does not contain v, then there is nothing to do and we simply set $W(i, H', \Phi', \ell') = W_j$. Otherwise, we consider the set Φ of all mappings φ that can be obtained from mappings φ' by removing v from its partition in the branch set. If v is the only vertex in its partition, then the corresponding vertex in F must additionally be mapped to $\perp$. We set $W(i, H, \Phi, \ell') = W_j$ where H is obtained from H' by removing v and mapping all edges from v to any vertex in X_i by a corresponding edge with the end-vertex corresponding to v in $\perp$. The resulting complexity of a forget node is asymptotically bounded by

$$\underbrace{2^{\binom{k}{2}}}_{\text{subgraphs } H} \cdot \underbrace{|V(F)|^{k+1} \cdot 2^{|V(F)|}}_{|\Phi'|} \cdot \underbrace{\mathcal{O}(1)}_{\text{remapping}} \cdot L$$

Join node i joining j_1 and j_2: Let j_1 and j_2 be the two children of i and let (j_1, H, Φ_1, ℓ_1) and (j_2, H, Φ_2, ℓ_2) be two interfaces. Two partial minor-embeddings $\varphi_1 \in \Phi_1$ and $\varphi_2 \in \Phi_2$ are compatible if all vertices $v \in X_{j_1} \cap X_{j_2}$ satisfy $\varphi_1^{-1}(v) = \varphi_2^{-1}(v)$. If φ_1 and φ_2 are compatible, we can obtain a new partial minor embedding by combining the two partial embeddings into a new partial minor embedding φ_{12}.

Let Φ_{12} be the set of partial minor embeddings combined in this manner from all pairs of compatible partial minor embeddings in $\Phi_1 \times \Phi_2$. Let $W_1 := W(j, H, \Phi_1, \ell_1)$ and $W_2 := W(j', H, \Phi_2, \ell_2)$. If $\ell := \ell_1 = \ell_2$ and $\Phi := \Phi_1 = \Phi_2$ we set $W(i, H, \Phi_1, \ell) = \max\{W_1, W_2\}$. Otherwise, we set $W(i, H, \Phi_1, \ell_1) = W_1$ and $W(i, H, \Phi_2, \ell) = W_2$. Additionally, we set $W(i, H, \Phi_1 \cup \Phi_2 \cup \Phi_{12}, \ell_1 + \ell_2 - \operatorname{len}(H)) := W_1 + W_2 - \operatorname{wt}(H)$. Clearly, the weight and length of H must be subtracted, since otherwise, these values would be counted twice.

Since $|\Phi_1 \times \Phi_2|$ is bounded by $|V(F)|^{k+1} \times |V(F)|^{k+1}$, the resulting complexity in total is asymptotically bounded by

$$2^{\binom{k}{2}} \cdot |V(F)|^{2k+2} \cdot 2^{2|V(F)|} \cdot 2^k \cdot |V(F)| \cdot k^2 \cdot M \cdot n = 2^{\mathcal{O}(k^2 + k \log |V(F)| + |V(F)|)} Mn \ .$$

If $\mathcal{F}$ contains more than one obstruction, the running time can be bounded by

$$2^{\mathcal{O}(k^2 + k \log N + N)} |\mathcal{F}| Ln \ ,$$

where N denotes the maximum number of vertices of any graph in $\mathcal{F}$. □

The following result can be obtained by a straightforward modification of the approach sketched in this section. With the technique developed in the next section, this result will yield an FPTAS for the Relaxed Maximum Density Subgraph problem.

Corollary 5.1. *Let $(G, \operatorname{wt}, \operatorname{len}, W)$ be an instance of the* RMDS *and let G and $\mathcal{F}$ be as in Theorem 5.4. Then for any $\lambda \in \mathbb{R}$ a maximum penalized density $\mathcal{F}$-minor-free (W, λ)-viable pattern can be computed in time $\mathcal{O}(2^{\mathcal{O}(k^2 + k \log N + N)} |\mathcal{F}| \lambda n)$ where $N = \max_{F \in \mathcal{F}} |V(F)|$.*

5.3 An FPTAS for Relaxed Density Maximization

In this section we consider the Relaxed Maximum Density Subgraph (RMDS) problem, where the upper bound on the length may be violated at the cost of an additional penalty term. We assume that the weight function is strictly positive. While it is $\mathcal{NP}$-hard to decide whether a feasible solution exists for the original problem, we show that this slight relaxation allows us to give an FPTAS for penalized density. This FPTAS can be applied to any problem that allows a quasi-polynomial-time algorithm that computes an optimal solution with respect to the penalized density. Note that the relaxed density maximization problem remains NP-hard as we can choose L very small such that every subgraph is penalized and we have that $\mathrm{pdens}(H) \approx \mathrm{dens}(H)/2$ for any subgraph H. Then the NP-hardness result of Theorem 5.1 naturally applies to this problem. For simplicity, we will assume that the scaling constant c for the penalized density equals 1.

Let Π be a relaxed density maximization problem that admits an algorithm $\mathcal{A}$ that takes as input an instance I of the relaxed density maximization problem and $\lambda \in \mathbb{N}$ and computes an optimal (W, λ)-viable pattern H with respect to penalized density, pdens, in $\mathcal{O}(p(\lambda, n))$ time, where $p(\lambda, n)$ is a function that is polynomial in λ and n. We show how to construct an FPTAS for Π that uses $\mathcal{A}$ as a subroutine. We present our algorithm within the terminology introduced by Schuurman and Woeginger for approximation schemes [SW]. We first structure the output of our algorithm to form exponentially growing buckets based on the length of the solutions. In order to compute approximately optimal solutions in each of the buckets efficiently, we structure the input of algorithm $\mathcal{A}$ by exponentially compressing the lengths and weights in such a way that the error resulting from the compression is proportional to the size of the solutions in each bucket.

Assume that we are given a graph T. Let k be a suitably chosen integer depending on ε, which will be defined later. We structure the output in $\lfloor \log_k B \rfloor - 1$ buckets, where $B = \mathrm{len}(G)$, such that bucket i with $0 \leq i \leq \lfloor \log_k B \rfloor - 2$ contains solutions with total length at most $k^{i+2}m$, where m is the number of edges. For each bucket we compute an approximately optimal solution and return the overall best solution as output of our algorithm. To compute an approximately optimal solution for bucket i, we structure the input by considering instances $I_i = (G, \mathrm{len}_i, \mathrm{wt}_i, W_i, L_i)$, where $\mathrm{len}_i(e) = \lceil \mathrm{len}(e)/k^i \rceil$, $\mathrm{wt}_i(e) = \mathrm{wt}(e)/k^i$ for $e \in E(G)$ and $W_i = W/k^i$ as well as $L_i = L/k^i$. We can think of these instances as being *compressed*. We apply algorithm $\mathcal{A}$ on the compressed instances I_i with $\lambda = k^2 m$. A high-level description of this algorithm is listed as Algorithm 5.1. When considering the i-th bucket, we refer to the deviation of $H \subseteq G$ with respect to len_i and L_i as the *compressed deviation* $\mathrm{dev}_i(H) = \max\{0, \mathrm{len}_i(H) - L_i\}$. Similarly, the penalized density of $H \subseteq G$ is defined as the *compressed penalized density* $\mathrm{pdens}_i(H) = \mathrm{wt}_i(H)/(\mathrm{len}_i(H) + \mathrm{dev}_i(H))$.

In order to show that Algorithm 5.1 is an FPTAS we proceed in several steps. First, we bound the compressed penalized density used in the i-th iteration of the algorithm in terms of the ordinary penalized density. In Lemma 5.4 we use this bound to derive an approximation ratio for the penalized density. Finally, we show that the algorithm is an FPTAS in Theorem 5.5.

Algorithm 5.1: FPTAS for Relaxed Density Maximization

```
Input: An instance $(G, \mathrm{wt}, \mathrm{len}, W, L)$ of Relaxed Maximum Density Subgraph, a
       real number $0 < \epsilon < 1$
Output: An $(1-\epsilon)$-approximation of the maximum penalized density subgraph
1 $k \leftarrow \lceil \frac{2}{\epsilon} \rceil$
2 for $i \leftarrow 0$ to $\lfloor \log_k B \rfloor - 1$ do
3     $H_i \leftarrow$ result of $\mathcal{A}$ on instance $I_i$ with $\lambda = k^2 m$
4 return $\max_{0 \le i \le \lfloor \log_k B \rfloor} \mathrm{pdens}_i(H_i)$
```

Lemma 5.3. *For any subgraph H and each $1 \le i < \lfloor \log_k B \rfloor$, the following holds*

$$\begin{aligned} \mathrm{len}(H) &\le k^i \cdot \mathrm{len}_i(H) &&\le \mathrm{len}(H) + |E(H)| \cdot k^i, && (5.1)\\ \mathrm{dev}(H) &\le k^i \cdot \mathrm{dev}_i(H) &&\le \mathrm{dev}(H) + |E(H)| \cdot k^i, && (5.2)\\ \mathrm{pdens}_i(H) &\le \mathrm{pdens}_{i-1}(H) &&\le \mathrm{pdens}(H), && (5.3)\\ \mathrm{len}_i(H) \le k \cdot m & \quad \textit{implies that} && \mathrm{len}_{i-1}(H) \le k^2 \cdot m\,. && (5.4) \end{aligned}$$

Proof. We use the following equation, which holds for any real positive numbers $r, s \in \mathbb{R}$.

$$r \le s \cdot \left\lceil \frac{r}{s} \right\rceil \le r + s \qquad (\star)$$

We start out by proving Equation (5.1), which relates the length of a graph to the length of the corresponding subgraph in the compressed instance of iteration i. By the definition of len and Equation ($\star$) we have

$$\begin{aligned} \mathrm{len}(H) &= \sum_{e \in E(H)} \ell_e \\ &\le k^i \cdot \sum_{e \in E(H)} \left\lceil \frac{\ell_e}{k^i} \right\rceil && \text{by Equation } (\star) \\ &= k^i \,\mathrm{len}_i(H) \end{aligned}$$

On the other hand,

$$\begin{aligned} k^i \,\mathrm{len}_i(H) &= k^i \cdot \sum_{e \in E(H)} \left\lceil \frac{\ell_e}{k^i} \right\rceil \\ &\le \sum_{e \in E(H)} (\ell_e + k^i) && \text{by Equation } (\star) \\ &= \mathrm{len}(H) + |E(H)| \cdot k^i\,. \end{aligned}$$

Next, we consider Equation (5.2). Note that the first inequality trivially holds if $\mathrm{dev}(H) = 0$. So, we may assume that $\mathrm{dev}(H) > 0$. By applying Equation (5.1) and the definition of the compressed deviation we obtain

$$\mathrm{dev}(H) = \mathrm{len}(H) - L \le k^i \cdot \mathrm{len}_i(H) - L \le k^i \cdot \mathrm{dev}_i(H)\,.$$

The second inequality of Equation (5.2) again trivially holds if $\mathrm{dev}_i(H) = 0$. For $\mathrm{dev}_i(H) > 0$, we obtain

$$k^i \,\mathrm{dev}_i(H) = k^i \,\mathrm{len}_i(H) - L \leq \mathrm{len}(H) + |E(H)| \cdot k^i - L = \mathrm{dev}(H) + |E(H)| \cdot k^i$$

using Equation (5.1) and the definition of the compressed deviation.

Finally, we consider Equations (5.3) and (5.4). Note that the first inequality of Equation (5.3) implies the second inequality since $\mathrm{pdens}_0(H) = \mathrm{pdens}(H)$ holds for any subgraph H. From Inequality $(\star)$ we obtain that

$$\mathrm{len}_{i-1}(H) = \sum_{e \in E(H)} \left\lceil \frac{\ell_e}{k^{i-1}} \right\rceil \leq \sum_{e \in E(H)} k \cdot \left\lceil \frac{\ell_e}{k \cdot k^{i-1}} \right\rceil = k \cdot \mathrm{len}_i(H)$$

for any subgraph H, which immediately implies Equation (5.4). Similarly, we also obtain

$$k \cdot \mathrm{dev}_i(H) = k \cdot \max\{0, \mathrm{len}_i(H) - L\} \geq k \cdot \max\{0, \mathrm{len}(H) - L\} = \mathrm{dev}_{i-1}(H)$$

Therefore, we obtain

$$\mathrm{pdens}_i(H) = \frac{\mathrm{wt}_{i-1}(H)}{k \cdot (\mathrm{len}_i(H) + \mathrm{dev}_i(H))} \leq \frac{\mathrm{wt}_{i-1}(H)}{\mathrm{len}_{i-1}(H) + \mathrm{dev}_{i-1}(H)} = \mathrm{pdens}_{i-1}(H)$$

using $\mathrm{wt}_i(H) = \mathrm{wt}_{i-1}(H)$ and $\mathrm{len}_i(H) + \mathrm{dev}_i(H) = k \cdot (\mathrm{len}_i(H) + \mathrm{dev}_i(H))$, which concludes the proof. □

Let $\Omega(H)$ be the smallest integer such that $\mathrm{len}_{\Omega(H)}(H) \leq k^2 m$. In other words, $\Omega(H)$ denotes the smallest bucket for which H will be considered by algorithm $\mathcal{A}$. Equation (5.4) immediately implies a lower bound on the length of H in this bucket.

Corollary 5.2. *For any subgraph H, $\Omega(H) > 0$ implies $\mathrm{len}_{\Omega(H)}(H) > km$.*

Now we are ready to bound the penalized density of an instance H in bucket $\Omega(H)$ in terms of k and its true penalized density $\mathrm{pdens}(H)$.

Lemma 5.4. *For any subgraph H, we have $\mathrm{pdens}_{\Omega(H)}(H) \geq \frac{k-1}{k+1} \cdot \mathrm{pdens}(H)$.*

Proof. Clearly, this inequality holds if $\Omega(H) = 0$. For $\Omega(H) \geq 1$, we have

$$\begin{aligned}
\mathrm{len}(H) &\geq k^{\Omega(H)} \cdot \mathrm{len}_{\Omega(H)}(H) - |E(H)| \cdot k^{\Omega(H)} && \text{by Equation (5.1)} \\
&\geq k^{\Omega(H)} \cdot (\mathrm{len}_{\Omega(H)}(H) - m) && \text{since } |E(H)| \leq m. \\
&\geq k^{\Omega(H)} \cdot (km - m) && \text{due to Corollary 5.2} \\
&= (k-1) \cdot k^{\Omega(H)} m \,. && (5.5)
\end{aligned}$$

This implies

$$k^{\Omega(H)} m \leq \frac{\mathrm{len}(H)}{k-1} \leq \frac{\mathrm{len}(H) + \mathrm{dev}(H)}{k-1} \,. \tag{5.6}$$

Then the penalized density satisfies

$$\begin{aligned}
\operatorname{pdens}_{\Omega(H)}(H) &= \frac{\operatorname{wt}_{\Omega(H)}(H)}{\operatorname{len}_{\Omega(H)}(H) + \operatorname{dev}_{\Omega(H)}(H)} && \text{by definition of pdens} \\
&= \frac{\operatorname{wt}(H)}{k^{\Omega(H)} \cdot (\operatorname{len}_{\Omega(H)}(H) + \operatorname{dev}_{\Omega(H)}(H))} && \text{by definition of } \operatorname{wt}_{\Omega(H)} \\
&\geq \frac{\operatorname{wt}(H)}{\operatorname{len}(H) + \operatorname{dev}(H) + 2|E| \cdot k^{\Omega(H)}} && \text{by Equations (5.1) and (5.2)} \\
&\geq \frac{\operatorname{wt}(H)}{\operatorname{len}(H) + \operatorname{dev}(H) + 2m \cdot k^{\Omega(H)}} && \text{since } |E| \leq m \\
&\geq \frac{\operatorname{wt}(H)}{\left(1 + \frac{2}{k-1}\right) (\operatorname{len}(H) + \operatorname{dev}(H))} && \text{by Equation (5.6)} \\
&= \frac{k-1}{k+1} \cdot \operatorname{pdens}(H) \ .
\end{aligned}$$

This concludes the proof. □

Theorem 5.5. *Given $0 < \varepsilon < 1$, we can compute a $(1-\varepsilon)$-approximation for the relaxed density maximization problem in $\mathcal{O}(p(m/\varepsilon^2, n) \log B)$ time, where G is the input graph and B is the maximum total length of the edges, provided that an $\mathcal{O}(p(\lambda, n))$ time algorithm for the penalized density maximization as described above exists.*

Proof. Clearly, the algorithm computes a W-viable solution if one exists due to the correctness of algorithm $\mathcal{A}$, the fact that we do not introduce any errors when scaling the weights, and since the union of the buckets covers all feasible solutions.

Next we show that the algorithm indeed produces a $(1-\varepsilon)$ approximation of the optimal penalized density. Let opt be an optimal solution and H^* be the solution returned by our algorithm. By the above lemmas and choosing $k = \left\lceil \frac{2}{\varepsilon} \right\rceil$, we have

$$\begin{aligned}
\operatorname{pdens}(H^*) &\geq \max_{i \geq \Omega(H^*)} \operatorname{pdens}_i(H^*) && \text{by Algorithm 5.1} \\
&\geq \max_{i \geq \Omega(\text{opt})} \operatorname{pdens}_i(\text{opt}) && \text{by Algorithm 5.1} \\
&\geq \operatorname{pdens}_{\Omega(\text{opt})}(\text{opt}) && \text{by Equation (5.3)} \\
&\geq \left(\frac{k-1}{k+1}\right) \cdot \operatorname{pdens}(\text{opt}) && \text{by Lemma 5.4} \\
&= \left(1 - \frac{2}{k+1}\right) \cdot \operatorname{pdens}(\text{opt}) \\
&\geq (1-\varepsilon) \cdot \operatorname{pdens}(\text{opt}) && \text{by definition of } k.
\end{aligned}$$

The running time of this approach clearly is $\mathcal{O}(p(m/\varepsilon^2, n) \log B)$ since $k = \left\lceil \frac{2}{\varepsilon} \right\rceil \geq 2$, and $\log_k B \leq \log_2 B = \mathcal{O}(\log B)$. □

For reasons of simplicity, we assumed a scaling factor $c = 1$. By choosing $k = \lceil c + 1/\varepsilon \rceil$ we can accomplish the same result for any scaling factor $c \neq 1$. In our analysis, we further assumed that we are given an algorithm $\mathcal{A}$ that computes a (W, λ)-viable pattern for a given value of λ.

However, our approach still works if $\mathcal{A}$ only computes a W-viable pattern with maximum penalized density. In each iteration we pre-process the instance I_i by removing edges that are longer than k^2m from G. Then the maximum length of any W-viable pattern considered by $\mathcal{A}$ is naturally bounded by k^2m^2. The running time of the resulting FPTAS is bounded by $\mathcal{O}(p(m^2/\varepsilon^2, n)\log B)$, assuming that $\mathcal{A}$ has a running time bounded by $\mathcal{O}(p(\mathrm{len}(G), n))$. Finally, with the results from Corollary 5.1 we immediately obtain the following result as an application of the FPTAS to the problem of maximizing the penalized density objective function.

Corollary 5.3. *Let $(G, \mathrm{wt}, \mathrm{len}, W)$ be an instance of the* RMDS *problem such that G has treewidth at most k and let $\mathcal{F}$ be a finite set of graphs. Let $B := \mathrm{len}(G)$, $0 < \varepsilon < 1$ and let* opt *be the optimal penalized density of an $\mathcal{F}$-minor-free W-viable pattern. Then a W-viable $\mathcal{F}$-minor-free pattern with penalized density at least $(1-\varepsilon)\cdot \mathrm{opt}$ can be computed in time $\mathcal{O}(2^{\mathcal{O}(k^2+k\log N+N)}|\mathcal{F}|m/\varepsilon^2\log B)$.*

5.4 Maximum Density Subgraphs with Structural Constraints

In this section we drop the lower bound on the weight as well as the upper bound on the length. Instead we impose structural constraints on the set of vertices by requiring a subset of the vertices to be contained in any feasible solution. This models a scenario in which we would like to inter-connect a subset of the given vertices in a certain way. For instance, we may wish to connect the sites to form a spanning tree or a perfect matching or we may wish to inter-connect a (small) subset of the vertices.

5.4.1 Parametric Search and Application

One of the most natural constraints on the vertex set is to require the pattern to span the whole set of vertices. Chandrasekaran [Cha77] shows that a spanning tree with maximum density can be computed in polynomial time. We provide an adapted version of the main theorem that makes this possible, along with a proof of its correctness, and show how this can be used as a generic tool in order to obtain efficient algorithms for the maximum density subgraph problem.

Theorem 5.6 (Chandrasekaran [Cha77])**.** *Let $G = (V, E)$ be a graph with edge weights $a_e \in \mathbb{Z}$, $b_e \in \mathbb{N}$ for $e \in E$ such that $b_e > 0$ for all $e \in E$ and let $\mathcal{S} \in 2^E$ be an arbitrary collection of subsets of the edges. Let*

$$\theta^* = \max_{X\in\mathcal{S}}\left\{\frac{\sum_{e\in X} a_e}{\sum_{e\in X} b_e}\right\} \quad \textit{and} \quad \varphi(\theta) = \max_{X\in\mathcal{S}}\left\{\sum_{e\in X}(a_e - \theta b_e)\right\}\ ,$$

then

$$\varphi(\theta) = \begin{cases} >0 & \Leftrightarrow \quad \theta < \theta^* \\ =0 & \Leftrightarrow \quad \theta = \theta^* \\ <0 & \Leftrightarrow \quad \theta > \theta^* \end{cases}\ .$$

Proof. Let $X^* \in \mathcal{S}$ be such that $\varphi(\theta) = \sum_{e \in X^*}(a_e - \theta \cdot b_e)$:

$$\varphi(\theta) = \sum_{e \in X^*} (a_e - \theta \cdot b_e) < 0 \tag{5.7}$$

$$\Leftrightarrow \quad \sum_{e \in X} (a_e - \theta \cdot b_e) < 0 \qquad \forall X \in \mathcal{S} \tag{5.8}$$

$$\Leftrightarrow \quad \frac{\sum_{e \in X} a_e}{\sum_{e \in X} b_e} < \theta \qquad \forall X \in \mathcal{S} \tag{5.9}$$

$$\Leftrightarrow \quad \theta^* < \theta \tag{5.10}$$

On the other hand, assume $\theta^* > \theta$. Then and only then there is some $X' \in \mathcal{S}$ such that

$$\frac{\sum_{e \in X'} a_e}{\sum_{e \in X'} b_e} > \theta \tag{5.11}$$

$$\Leftrightarrow \quad \exists X' \in \mathcal{S} \ : \ \sum_{e \in X'} (a_e - \theta \cdot b_e) > 0 \tag{5.12}$$

$$\Leftrightarrow \quad \varphi(\theta) = \sum_{e \in X^*} (a_e - \theta \cdot b_e) > 0 \; . \tag{5.13}$$

Equality for the case $\varphi(\theta) = 0$ follows from the previous observations. □

In order to solve maximum density subgraph problems we adapt the optimization algorithm suggested by Chandrasekaran to our setting. Essentially, the algorithm performs binary

Algorithm 5.2: Parametric Search

```
Input: Graph $G = (V, E)$, $a_e \in \mathbb{Z}$, $b_e \in \mathbb{N}$ for $e \in E$, set of feasible solutions $\mathcal{S} \subseteq 2^E$
Output: $S^* \subseteq \mathcal{S}$ with maximum density
1  $[\alpha, \beta] \leftarrow \left[\min_{e \in E}\left\{\frac{a_e}{b_e}\right\}, \max_{e \in E}\left\{\frac{a_e}{b_e}\right\}\right]$
2  while $\beta - \alpha \geq \left(\sum_{e \in E} b_2\right)^{-2}$ do
3      $k \leftarrow \frac{\alpha+\beta}{2}$
4      $d \leftarrow \max_{X \in \mathcal{S}}\left\{\sum_{e \in X}(a_e - k \cdot b_e)\right\}$
5      if $d > 0$ then
6          $[\alpha, \beta] \leftarrow [k, \beta]$
7      else if $d < 0$ then
8          $[\alpha, \beta] \leftarrow [\alpha, k]$
9      else
10         $[\alpha, \beta] \leftarrow [k, k]$
11 $S_\alpha \leftarrow \operatorname{argmax}_{X \in \mathcal{S}}\left\{\sum_{e \in X}(a_e - \alpha \cdot b_e)\right\}$
12 $S_\beta \leftarrow \operatorname{argmax}_{X \in \mathcal{S}}\left\{\sum_{e \in X}(a_e - \beta \cdot b_e)\right\}$
13 if $\operatorname{dens}(S_\alpha) > \operatorname{dens}(S_\beta)$ then
14     return $S_\alpha$
15 else
16     return $S_\beta$
```

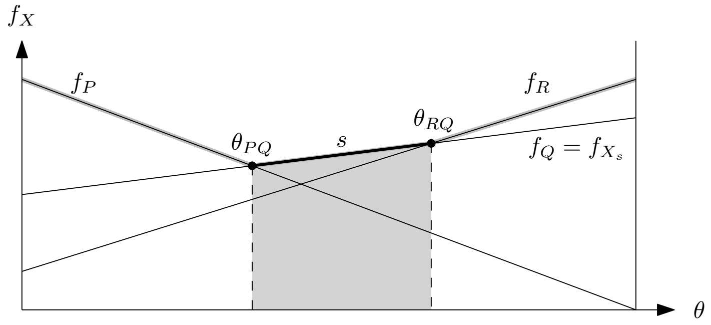


Figure 5.4: Upper boundary of the set of functions $\mathcal{F}_\mathcal{S}$ (gray line), segment s on the upper boundary (bold black segment) and corresponding linear function f_{X_s} associated with $X_s \in \mathcal{S}$ and interval that is dominated by s (gray area).

search on the density space using Theorem 5.6 and is listed as Algorithm 5.2. In Lemma 5.5 we show that this algorithm finds an optimal solution after a polynomial number of steps, when the interval containing the optimal density is smaller than some value depending on the input numbers. In this case, we return the maximum density of the two solutions corresponding to the interval boundaries.

Lemma 5.5. *Let $G = (V, E)$ be a graph with edge weights $a_e \in \mathbb{Z}, b_e \in \mathbb{N}$ for $e \in E$ and let $\mathcal{S} \subseteq 2^E$ be a set of feasible solutions. If we can compute*

$$\text{opt}(\theta) = \text{argmax}_{X \in \mathcal{S}} \left\{ \sum_{e \in X} (a_e - \theta b_e) \right\}$$

in time $f(n)$ then Algorithm 5.2 computes a solution $X^* \in \mathcal{S}$ with maximum density in time $\mathcal{O}(f(n) \log(nM))$ where M denotes the largest input number.

Proof. First, we show that, if an interval I containing the optimal density is smaller than some threshold value depending on the input numbers, then the optimal solution corresponds to a solution that can be associated with the densities at the boundaries of the interval. Note that every $X \in \mathcal{S}$ corresponds to a linear function $f_X : \theta \mapsto \sum_{e \in X} a_e - \theta \cdot \sum_{e \in X} b_e$. Essentially we are interested in examining the upper envelope of the set of functions $\mathcal{F}_\mathcal{S} := \{f_X \mid X \in \mathcal{S}\}$. This upper boundary is composed of linear segments, such that each segment s corresponds to some $X_s \in \mathcal{S}$ and to some interval I_s on the θ-axis. This interval contains all θ for which X_s maximizes $\sum_{e \in X}(a_e - \theta \cdot b_e)$ over all $X \in \mathcal{S}$. We say that X_s is *dominating* on I_s. See Figure 5.4 for an illustration.

Let $f_P, f_Q, f_R, f_S \in \mathcal{F}_\mathcal{S}$ and let θ_{PQ} and θ_{RS} be the intersections of f_P, f_Q and f_R, f_S, respectively, such that $\theta_{PQ} < \theta_{RS}$. Then the length of the interval $[\theta_{PQ}, \theta_{RS}]$ can be bounded from below as follows. Let $A_X := \sum_{e \in X} a_e$ and let $B_X := \sum_{e \in X} b_e$. Then we have

$$f_P(\theta_{PQ}) = A_P + \theta_{PQ} B_P = A_Q + \theta_{PQ} B_Q = f_Q(\theta_{PQ})$$
$$f_R(\theta_{RS}) = A_R + \theta_{RS} B_R = A_S + \theta_{RS} B_S = f_S(\theta_{RS})$$

which, by a simple transformation, is equivalent to

$$\theta_{PQ} = \frac{A_P - A_Q}{B_Q - B_P} \quad \text{and} \quad \theta_{RS} = \frac{A_R - A_S}{B_S - B_R} \, .$$

It follows that, if $|\theta_{PQ} - \theta_{RS}| > 0$, we have

$$\begin{aligned}|\theta_{PQ} - \theta_{RS}| &= \left|\frac{A_P - A_Q}{B_Q - B_P} - \frac{A_R - A_S}{B_S - B_R}\right| \\ &= \left|\frac{(A_P - A_Q)(B_S - B_R) - (A_R - A_S)(B_Q - B_P)}{(B_Q - B_P)(B_S - B_R)}\right| \\ &\geq \left|\frac{1}{(B_Q - B_P)(B_S - B_R)}\right|\end{aligned}$$

since the numerator must be an integer

$$\geq \frac{1}{\left(\sum_{e\in E} b_e\right)^2}$$

since both B_Q and B_S are bounded by $\sum_{e\in E} b_e$. Hence, whenever $|\theta_{RS} - \theta_{PQ}| > 0$, we have

$$|\theta_{RS} - \theta_{PQ}| > \left(\sum_{e\in E} b_e\right)^{-2} .$$

As a consequence, let $[\alpha, \beta]$ be an arbitrary non-degenerate interval of length less than this value containing the optimal density θ^*. Then this interval may contain at most one intersection point of all the pairs of linear functions $f_P, f_Q \in \mathcal{F}_\mathcal{S}$, that is, it intersects at most two segments on the upper boundary of $\mathcal{F}_\mathcal{S}$. Then, clearly, either the solution $\text{opt}(\alpha)$ corresponding to α or the solution $\text{opt}(\beta)$ corresponding to β must be optimal. Note, that the optimal density will, in general, match neither α nor β in this case.

Next, we prove the bound on the running time. It is clear that $\delta_{\min} \leq \theta^* \leq \delta_{\max}$, where $\delta_{\min}$ and $\delta_{\max}$ denote the minimum and maximum density of an edge of G, respectively. Hence we only need to search the optimal value in the interval $[\delta_{\min}, \delta_{\max}]$. The size of this interval is at most $|2a_{\max}|$, where $a_{\max}$ denotes the maximum weight of an edge. since we assumed $b_e \geq 1$ for all $e \in E$. Algorithm 5.2 performs binary parameter search on this interval using Theorem 5.6. In each step we bisect the previous interval, that is, after t steps the interval has size at most $2|a_{\max}|/2^t$. Thus, after $t > \log|a_{\max}| + 2 \cdot \log \sum_{e\in E} b_e + 1$ steps the size of the interval is smaller than $\left(\sum_{e\in E} b_e\right)^{-2}$. By previous arguments either the solution corresponding to the left boundary of the interval or the solution corresponding to the right boundary of the interval is an optimal solution. Hence, it suffices to compute two optimal solutions for α and β, respectively, and to compare their densities. The running time of the algorithm is in $\mathcal{O}(f(n) \cdot \log(nM))$ where M is the largest absolute value of the input numbers. Hence, the running time is polynomial in the input size. □

The algorithm provides a generic tool that can be applied to various problems, whenever the corresponding single-objective optimization problem can be solved efficiently in the presence of both positive and negative numbers. For instance, since perfect weighted matchings can be computed in time $\mathcal{O}((m + n\log n)n)$ [Gab90] the lemma immediately implies the following.

Corollary 5.4. *A perfect maximum density matching can be computed in time $\mathcal{O}((m + n\log n)n\log(nM))$.*

Next, we show that a maximum density subtree with k leaves can be computed in a tree in polynomial time, again using the fact that we can solve the underlying single-objective optimization problem efficiently.

Theorem 5.7. *Given a tree $T = (V, E)$, a maximum density subtree with exactly k leaves can be computed in time $\mathcal{O}(k^2 n \log(nM))$ where M denotes the largest input number.*

Proof. The proof exploits a combination of the parametric search technique and dynamic programming. By applying the parametric search we reduce the problem to finding—for various values of $\theta \in \mathbb{R}$—a longest subtree of T with k leaves, where the new length of each edge e is given by $a_e - \theta b_e$. For an edge $e = \{u, v\}$ we denote this new length by $\lambda(u, v) = a_e - \theta b_e$.

In order to do compute the longest subtree of T with k leaves, we root the tree in some vertex $r \in V$. For a vertex $v \in V$ we denote the number of children of v by $n(v)$ and we denote the children by $u_1^v \dots, v_{n(v)}^v$.

Suppose we are given an optimal solution T^* for this problem. Let v^* be the topmost vertex in T^* with respect to the rooting. Then either v^* is a leaf in T^* and there are only $k-1$ leaves of T^* in the subtrees rooted in the children of v or v^* is an internal vertex with k leaves in the subtrees rooted in the children of v^*. Let $u \in V$ be a vertex of T and let T_u denote the subtree of T rooted in u. Let T'_u be a subtree of T_u containing u. Further, let T^*_u denote the subtree of T^* that is contained in T'_u and let k' denote the number of leaves of T^* contained in T'_u. Then, clearly T^*_u is the longest subtree in T'_u rooted in u with k' leaves. We use this observation to decompose the problem into smaller sub-problems.

Let $v \in V$, $1 \le i \le k-1$ and $1 \le j \le n(v)$. Then we denote by $\Lambda(v, i, j)$ the length of a longest tree $\widehat{T}$ with i leaves, where v does not count as a leaf, such that $\widehat{T}$ is contained in the tree induced by v and the subtrees rooted in its children $u_1^v, \dots, u_j^v$. We can compute these values from the following equation

$$\Lambda(v, i, j+1) = \max \left\{ \begin{array}{l} \Lambda(v, i, j), \\ \Lambda(v, i-1, j) + \lambda(v, u_{j+1}), \\ \max_{1 \le t \le i} \{\Lambda(v, i-t, j) + \Lambda(u_{j+1}, t, n(u_{j+1}))\} + \lambda(v, u_{j+1}) \end{array} \right\}$$

in a bottom-up manner on the tree, using $\Lambda(v, 0, 0) = 0$. This can be done in $\mathcal{O}(k^2 n)$ time.

Further, we denote by $\overline{\Lambda}(v, i, j)$ the length of a longest tree $\widehat{T}$ with i leaves, where v does not count as a leaf, such that $\widehat{T}$ is contained in the tree induced by v and exactly one subtree rooted in some child u_r^v of v, where $j \le r \le n(v)$. These values can be computed in $\mathcal{O}(kn)$ time from the following equation using the values computed in the previous step

$$\overline{\Lambda}(v, i, j) = \max_{j \le r \le n(v)} \{\Lambda(u_r, i, n(u_r)) + \lambda(v, u_r)\} .$$

Clearly, a tree with k leaves can only be found in a subtree of T with at least $k-1$ leaves. For each vertex v such that the tree T_v rooted in v contains at least $k-1$ leaves, we compute the longest subtree with k leaves, denoted by $\Lambda^*(v)$, as follows

$$\Lambda^*(v) = \max \left\{ \begin{array}{l} \overline{\Lambda}(v, k-1, 1), \\ \max_{1 \le t \le k-1, 1 \le r \le n(v)} \{\Lambda(v, k-t, r) + \overline{\Lambda}(v, t, r+1)\} \end{array} \right\} .$$

As mentioned earlier, this equation reflects the fact that v is either a leaf itself or an internal vertex, in which case v must have at least two children in T_v. We achieve this by “guessing” an index r such that the computed tree contains at least on child in $u_1^v, \dots, u_r^v$ and one child in $u_{r+1}^v, \dots, u_{n(v)}^v$. Again, these values can be computed in $\mathcal{O}(kn)$ time. Finally, we return the solution corresponding to the maximum value $\Lambda * (v)$ over all $v \in V$ with at least $k-1$ leaves in the subtree T_v. Using Lemma 5.5 the total time complexity is $\mathcal{O}(k^2 n \log(nM))$. □

5.4.2 Maximum Density Subgraphs Spanning a Subset of the Vertices

In this section we consider maximum density subgraph problems with Steiner constraints. Given a graph $G = (V, E)$ and a set of *terminals* $S \subseteq V$ the MAXIMUM DENSITY STEINER SUBGRAPH problem asks for a maximum density subgraph H containing all vertices in S. First we show that this problem NP-hard and inapproximable unless $\mathcal{P} = \mathcal{NP}$, even if the pattern is a path and there is only one terminal.

Theorem 5.8. *The* MAXIMUM DENSITY STEINER SUBGRAPH *problem is $\mathcal{NP}$-hard, even if all weights are positive, all numbers are chosen from two distinct values, there is only one terminal and the pattern is a path. Furthermore, unless $\mathcal{P} = \mathcal{NP}$, this problem can not be approximated within a constant factor in polynomial time under the same conditions.*

We prove this theorem by a two-step reduction from the LONGEST PATH problem. Given a graph $G = (V, E)$ the LONGEST PATH problem is to compute a path with maximum length, where the length is given by the number of edges on this path. This problem is NP-hard [GJ79] and cannot be approximated within a constant factor unless $\mathcal{P} = \mathcal{NP}$ due to Karger et al. [KMR97]. First, we show that this problem remains NP-hard and inapproximable if we require that the path starts in a predefined vertex $r \in V$. We refer to this problem as the ROOTED LONGEST PATH problem and we refer to r as the root.

Lemma 5.6. *The* ROOTED LONGEST PATH *problem is $\mathcal{NP}$-hard and cannot be approximated within a constant factor unless $\mathcal{P} = \mathcal{NP}$.*

Proof. Let $\mathcal{A}$ be an algorithm that approximates the ROOTED LONGEST PATH problem within a factor r. Then we immediately obtain an algorithm $\mathcal{A}'$ approximating the longest path by running $\mathcal{A}$ with root v once for each $v \in V$ and returning the maximum of these values. Clearly, $\mathcal{A}'$ approximates the longest path within a factor of r. The claim then follows from the results of Karger et al. [KMR97]. □

Proof of Theorem 5.8. We make a reduction from the ROOTED LONGEST PATH problem. Assume we are given an instance $I = (G, r)$ of the ROOTED LONGEST PATH problem, where $G = (V, E)$ is a graph and $r \in V$ is the root. We construct a new instance $I' = (G', S)$ of the MAXIMUM DENSITY STEINER SUBGRAPH problem as follows. We let $G' = (V', E')$ such that $V' = V \cup \{x\}$ for some new vertex $x \notin V$ and we set $E' = E \cup \{\{x, r\}\}$ and $S = \{x\}$. Further, let $M := n^2 + 1$. We set $w_e = 1$ and $\ell_e = M$ for $e = \{x, r\}$ and we set $w_e = M$ and $\ell_e = 1$ for all $e \in E$.

We claim that there is a path in G' rooted in x with density at least θ if and only if there is a path in G with length at least $\lceil \theta - 1 \rceil$ rooted in r. Note that a path in G' of length $i+1$ rooted in x has density $\theta_i = (M + iM)/(M + i)$ for $i \geq 0$. Since θ_i is monotonically increasing in i and we have $i < \theta_i < i+1$ for $i \le n$ and $M \ge n^2$ it follows that $\lceil \theta_i - 1 \rceil = i$ and, hence, the claim holds.

Suppose that $\mathcal{A}$ is an approximation algorithm that approximates the Maximum Density Steiner Subgraph problem for path patterns within a factor $r_{\mathcal{A}}$. We show that we can use $\mathcal{A}$ to approximate the rooted longest path problem within a factor $3r_{\mathcal{A}}$. If r is isolated in G, we return r as a longest path, which is an optimal solution in this case. Otherwise, let e be an arbitrary edge incident to r in G. For a given instance $I = (G, r)$ let $P_{\mathcal{A}}$ be the path computed by $\mathcal{A}$ for the instance $I' = (G', S)$ constructed from I as described above and let $\theta_{\mathcal{A}}$ be its density. If $\theta_{\mathcal{A}} \ge 2M/(M+1)$ we return $P_{\mathcal{A}}$, otherwise we return the single edge e, which together with the edge $\{x, r\}$, forms a path of length two with density $2M/(M+1)$ in G'.

In the following, let opt_I denote the longest path in G rooted in r and let $\mathrm{opt}_{I'}$ denote the maximum density Steiner path in G'. Further, let $\mathrm{apx}_{I'}$ denote the density of the approximation as described above. Note that since $2M/(M+1) > 3/2$ for $n > 1$ we have $\mathrm{apx}_{I'} > 3/2$. It follows that

$$\begin{aligned}
\frac{\mathrm{opt}_I}{\mathrm{apx}_{I'}} &= \frac{\lceil \mathrm{opt}_{I'} - 1 \rceil}{\lceil \mathrm{apx}_{I'} - 1 \rceil} \\
&\le \frac{\mathrm{opt}_{I'}}{\mathrm{apx}_{I'} - 1} \\
&\le \frac{1}{1 - 1/\mathrm{apx}_{I'}} \cdot \frac{\mathrm{opt}_{I'}}{\mathrm{apx}_{I'}} \\
&< \frac{1}{1 - 2/3} r_{\mathcal{A}} \\
&= 3 \cdot r_{\mathcal{A}} .
\end{aligned}$$

The claim then follows from Lemma 5.6. □

Although the Maximum Density Steiner Subgraph problem is NP-hard and unlikely to be approximable if the pattern is a path, we may still be able to obtain fixed-parameter tractable algorithms. First, we show that it is unlikely that the general problem is FPT when parameterized by the number of vertices in the solution, when we have no constraint on the feasible patterns.

Theorem 5.9. Maximum Density Steiner Subgraph *is $W[1]$-hard when parameterized by the number of vertices of the Steiner subgraph, even if S contains only one vertex.*

Proof. We prove the theorem by reduction from the $W[1]$-hard problem k-Clique [DF95]. Given an instance of k-Clique, that is, a graph $G = (V, E)$, we wish to decide if G has a clique of size at least k. We transform this into an instance of Maximum Density Steiner Subgraph as follows. We construct a graph G' by adding a new vertex x to G that is connected to all vertices in V. We set $\mathrm{wt}_{vw} = \mathrm{len}_{vw} = 1$ for all $vw \in E$ and we set $\mathrm{wt}_{xv} = 0$ and $\mathrm{len}_{xy} = 1$ for all $v \in V$. Further, we set $S = \{x\}$. Clearly, there is a clique of size k with $m = \binom{k}{2}$ edges in G if and only if G' has a subgraph H with $x \in H$ and density at least $m/(m+1)$. □

While the Maximum Density Steiner Subgraph problem is W[1]-hard on general graphs, it turns out to be FPT on planar graphs.

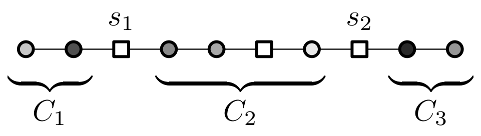


Figure 5.5: Decomposition of an optimal colorful Steiner path. Terminals are depicted as boxes.

Theorem 5.10. Maximum Density Steiner Subgraph *is FPT on planar graphs when parameterized by the number of vertices of the subgraph.*

Proof. Let G be a planar graph and let $S \subseteq V$ be a non-empty set of terminals. Since we are looking for a connected subgraph and since $S \neq \emptyset$, it suffices to consider the subgraph G' consisting of the $(k-1)$-neighborhood of some vertex $s \in S$. This graph is $(k-1)$-outerplanar and has radius at most $k-1$. Hence, by a result of Robertson and Seymour [RS84], G' has treewidth bounded by $3k-2$. Note, that a path is a simple, connected graph that does not contain a triangle as a minor. Hence, using a modification of the algorithm proposed in Theorem 5.4 in combination with Theorem 5.5, we can compute a subgraph of G' with maximum density in FPT time.

□

In contrast to the W[1]-hardness of the Maximum Density Steiner Subgraph problem without constraints on the pattern, we show that the problem is FPT when parameterized by the number of vertices if the pattern is a path. In order to obtain this result we use the parametric search technique introduced in Section 5.4.1 in combination with Color Coding [AYZ95]. Color Coding was introduced by Alon et al. [AYZ95] as a method for finding subgraphs of bounded size in arbitrary graphs in FPT time. It can be used to find paths and cycles in expected time $2^{\mathcal{O}(k)}m$. The algorithms are randomized by construction, but can be derandomized. We can directly apply Color Coding to find a maximum density Steiner path with at most k vertices with high probability (WHP) in time $2^{\mathcal{O}(k)}m \log(nM)$. We can slightly improve on the standard algorithm by coloring the terminals in a deterministic way, thus, using only $k-s$ random colors and thereby improving the probability of obtaining an optimal solution. In the following we further improve on this result when the number of Steiner vertices is large compared to k.

Theorem 5.11. *Given a set S of s terminals a maximum density Steiner path with k vertices can be computed WHP in time $\mathcal{O}((2^{k-s}m + 3^{k-s})s^2 \log(nM))$.*

Proof. Since we apply the parametric search technique, we need to find, for various values of θ, a longest path with respect to the edge-lengths $\lambda(e) := a_e - \theta b_e$ for all $e \in E$. Suppose we are given a random coloring of the vertices in $V \setminus S$ with $k-s$ colors using some color set C and we wish to find a longest *colorful* Steiner path for S. A Steiner path is called colorful, if it contains each color exactly once. Note, that this implies that the path is simple, that is, it does not contain multiple copies of any vertex. Our approach is based on a decomposition of any optimal colorful Steiner path P for S into three optimal subpaths P_1, P_2 and P_3 as illustrated in Figure 5.5 such that the following holds:

(i) P_2 starts and ends at terminals s_1 and s_2, respectively, and contains all terminals

(ii) P_1 and P_2 end at terminals s_1 and s_2, respectively, and do not contain any other terminals.

In order to compute P_2, we further decompose this path into fragments each starting and ending with a terminal and containing no other terminals apart from the end-vertices. Let $c(v)$ denote the color of vertex v. For each vertex $v \in V \setminus S$, $x \in S$ and each set of colors $C' \subseteq C$ with $c(v) \in C'$ we compute the maximum length $L(x, v, C')$ of any colorful path from x to v using all colors in C'. Similarly, we compute for each $z \in V$ the length $\widehat{L}(z, C')$ of the longest path ending in z using all colors in C'. We compute these values by dynamic programming in $\mathcal{O}(2^{k-s} \cdot s \cdot m)$ time using the equations

$$\begin{aligned} L(x, v, C') &= \max_{w \in N(v) \setminus S} \{L(x, w, C' \setminus \{c(v)\}) + \lambda(v, w)\} \\ \widehat{L}(z, C') &= \max_{w \in N(z) \setminus S} \{\widehat{L}(w, C' \setminus \{c(v)\}) + \lambda(w, z)\} \ . \end{aligned}$$

For each $x, y \in S$ and each set of colors $C' \subseteq C$ we subsequently compute the maximum length $L(x, y, C')$ of any path from x to y using all colors in C' as well as for all $z \in V$ the maximum length $\widehat{L}(z, C')$ of any path ending in z using all colors in C' by

$$L(x, y, C') = \max_{w \in N(y) \setminus S} \{L(x, w, C') + \lambda(w, y)\} \ .$$

in time $\mathcal{O}(2^{k-s} \cdot s^2 \cdot m)$. Implicitly, we obtain a combinatorial description of the longest paths starting and ending at terminals as a complete multigraph $G_S = (S, M)$ on the set of terminals in which each edge is annotated with its length and the set of colors used to attain this length. The weighted multiset of edges in our multigraph $G_S = (S, M)$ is defined by

$$M = \left\{ (xy, C', L(x, y, C')) \mid x, y \in S, C' \subseteq C \right\} \ .$$

Then we compute for each vertex $x \in S$ and each subset $X \subseteq S$ with $x \in X$ and each set of colors $C' \subseteq C$ the length $\widetilde{L}(x, X, C')$ of the longest path in G_S ending in x using all vertices in X and all colors in C' using the equation

$$\widetilde{L}(x, X, C') = \max_{y \in S, C'' \subseteq C'} \{\widetilde{L}(y, X \setminus \{x\}, C' \setminus C'') + L(x, y, C'')\}$$

where $L(x, \{x\}, C') = \widehat{L}(x, C')$. In order to do this efficiently, we consider all partitions of C into three sets $C \setminus C'$, C'' and $C' \setminus C''$. The number of these partitions is at most 3^{k-s}. Hence, the computation can be performed in time $\mathcal{O}(3^{k-s})s^2$. The optimal solution can then be computed as

$$\max_{x \in S, C' \subseteq C} \{\widetilde{L}(x, S, C') + \widehat{L}(x, C \setminus C')\} \tag{5.14}$$

in time $\mathcal{O}(2^{k-s}s)$. Hence, the dynamic programming for fixed θ takes $\mathcal{O}(2^{k-s}s^2m + 3^{k-s}s^2)$ time and the problem can be solved in time $\mathcal{O}((2^{k-s}m + 3^{k-s})s^2 \log(nM))$.

The probability that a path of length at most k interconnecting a set of s vertices is colorful is given by $p(k, s) = (k - s)!/(k - s)^{(k-s)} > \sqrt{2\pi(k - s)}e^{-(k-s)}$. Hence, a colorful path can be found with probability $\leq \varepsilon$ if the number of trials is at least $t_\varepsilon(k, s) = \frac{\ln \varepsilon}{\ln(1-p(k,s))} = |\ln \varepsilon| \cdot \mathcal{O}(e^{k-s})$.

□

We conclude this section by showing that we cannot hope to extend this result to more general patterns.

Theorem 5.12. *It is $\mathcal{NP}$-hard to decide whether there is a Steiner tree with at most k vertices and density at least θ, even if we allow only one terminal.*

Proof. We show $\mathcal{NP}$-hardness by reduction from the NP-hard k-MST problem. Given an edge-weighted graph G, a non-negative integer k and a weight W, the k-MST problem is to decide whether there is a tree spanning at least k vertices with weight at least W. This problem has been shown to be NP-hard by Ravi et al. [RSM+96] and it obviously remains NP-hard if we require the solution to contain exactly k vertices.

We observe that deciding whether there is a tree with k vertices and density at least θ is equivalent to deciding whether there is a tree with k vertices and length *at most* 0 where the length of each edge is given by $\theta b_e - a_e$. To see this, note that

$$\frac{\sum_{e\in E'} a_e}{\sum_{e\in E'} b_e} \geq \theta \Leftrightarrow \sum_{e\in E'} (\theta b_e - a_e) \leq 0$$

by simple equivalent transformations.

Assume we are given an instance of k-MST, that is, a graph $G = (V, E)$ and we wish to decide if there is a tree with k vertices and length at most θ where the edge-lengths are integral. We transform this into a set of $|V|$ k-Maximum Density Steiner Tree problems G_v such that G is solvable if and only if at least one of the new instances G_v is solvable. We choose $G_v = (V \cup \{x\}, E \cup \{xv\})$ for some new vertex x. Further, we choose the $b_e \equiv 1$ and $a_e := \ell_e - \theta$ and $a_{xv} = 2\theta$ and $S = \{x\}$. Hence, $\theta b_e - a_e = \ell_e$, $\theta b_{xv} - a_{xv} = -\theta$ and each solution contains the edge sx.

By the above observations there is a tree with k vertices and length at most θ if and only if there is a tree with $k+1$ vertices including x with length at most 0, where edge lengths are defined by $\theta b_e - a_e$. The latter is equivalent to deciding whether there is a tree with k vertices and density at least θ. $\square$

5.5 Concluding Remarks

We have investigated the complexity of a framework for bi-objective network design problems with one minimization and one maximization objective in the presence of additional constraints by studying the complexity of maximizing the ratio of the two objectives for different classes of hosts and patterns. Like many multi-objective optimization problems, the problems we considered turned out to be NP-hard in general. Nevertheless, we presented efficient algorithms for restricted variants of the problem as well as an FPTAS for a relaxed variant of the problem.

While the bi-constrained density maximization problem is already NP-hard if the host is an outerplanar graph, that is, a graph with treewidth two, and the pattern is a path, we were able to obtain a pseudo-polynomial time algorithm for the case when the host has bounded tree-width and the pattern is a graph from a minor-closed family of graphs. In some sense, this is the best we can hope for, since the existence of a polynomial time algorithm for this case would imply $\mathcal{P} = \mathcal{NP}$ as the class of paths clearly is a minor-closed family of graphs. Additionally, we presented an efficient algorithm for computing the maximum density path in a tree and in a graph that can be turned into a tree by removing only a fixed number of edges, respectively. Again, this is in some sense the best we can hope for. While graphs that can be turned into a tree by removing at most k edges have treewidth bounded by $k + 1$, we cannot hope to extend this result to the class of graphs with treewidth k unless $\mathcal{P} = \mathcal{NP}$. Interestingly, this result is obtained by applying geometric arguments to a problem that does not have any geometric context at first glance. Further, we showed that a relaxed variant of the density maximization problem admits a FPTAS, if we allow violating the upper bound on the length of the sought solution at some extra penalty. That is, for this relaxed variant, we can get rid of the pseudo-polynomiality at the cost of a loss of exactness.

Finally, we considered the density maximization problem in the presence of structural constraints on the vertex sets. Although some density maximization problems involving constraints on the vertex set of the graph, such as the problem of computing perfect matchings, can be solved efficiently by parametric search in the density-space, the general maximum density Steiner subgraph problem is NP-hard and inapproximable. Even worse, this problem is W[1]-hard when parameterized by the number of vertices of the sought solution. In contrast to this, we showed that the special case when the pattern is a path is fixed-parameter tractable when parameterized by this number. Again, we cannot hope to generalize this problem very much, since it turns out to be NP-hard already for general trees.

Open problems Due to the hard nature of both the bi-constrained density maximization problem and the relaxed density maximization problem, it seems unlikely that there exist efficient exact algorithms for all instances of the problem. Therefore we studied approximation algorithms and FPT algorithms. Whereas we only studied parameterizations with respect to treewidth and the number of vertices of the pattern, there may be other interesting parameters of the problem. Further, it seems to be an interesting challenge to devise efficient and effective heuristics for the problem. Another interesting problem concerns the question whether there is a meaningful class of graphs admitting the efficient computation of a maximum density path between the class of graphs with bounded tree-width and the class of graphs that can be turned into a tree by removing a fixed number of edges. Finally, it seems to be worthwhile to study variants of the problem in which we know more about the weights and the lengths of the edges, for instance, if we consider the problem on geometric graphs and the length corresponds to the Euclidean distance whereas the weight is proportional to the number of points in the vicinity of the edge, say. This problem variant can be motivated from network construction problems such as the construction of a new power supply system where each edge in the graph corresponds to a possible route for the network and points in the vicinity of an edge correspond to users that can profit from this line.

Chapter 6

Orthogeodesic Embedding of Planar Graphs

Up to this point we have considered purely combinatorial network construction problems. In this and the following chapters we turn our attention to geometric network construction problems such as the problems arising, for instance, in the context of very large scale integration circuit design. These network construction problems involve the computation of a geometric representation of the network—or an embedding of the network into the plane—and are typically considered in the field of graph drawing.

In this chapter and the following two chapters, we explore a new convention for embedding graphs into the plane, called the orthogeodesic or *Manhattan-geodesic* drawing convention. It requires that edges are drawn as interior-disjoint *monotone* chains of axis-parallel line segments, that is, as geodesics with respect to the Manhattan metric. From the perspective of geometric network construction the resulting networks will yield cost-effective and functional geometric representations of the considered networks since the network links are shortest possible and can easily be manufactured as a consequence of their orthogonal layout. From the graph drawing perspective, on the other hand, the resulting networks tend to yield very clear drawings, since the edges are short and monotone, and since the number of slopes is rather small.

First, we show that geodesic embeddability on the grid is equivalent to 1-bend embeddability on the grid. For the latter question an efficient algorithm has been proposed. Second, we consider *orthogeodesic point-set embeddability* where the task is to decide whether a given graph can be embedded on a given point set. We show that this problem is $\mathcal{NP}$-hard. In contrast, we efficiently solve *orthogeodesic polygonization*—the special case where the graph is a cycle. Third, we consider orthogeodesic point-set embeddability where the vertex–point correspondence is given. We show that, on the grid, this problem is $\mathcal{NP}$-hard even for perfect matchings. Without the grid restriction, however, we can efficiently test embeddability for any planar graph (of maximum degree 4). This chapter is based on joint work with Bastian Katz, Ignaz Rutter and Alexander Wolff [KKRW10, KKRW09].

6.1 Introduction

In this chapter we introduce a new drawing style for planar graphs. One of the most popular drawing styles is the *orthogonal* drawing style, which requires edges to be drawn as interior-disjoint *orthogonal* chains, that is, chains consisting of axis-parallel line segments. Restricting the number of edge directions—as in the rectilinear drawing style—potentially yields very clear drawings. We go a step further and insist that, additionally, edges are drawn as *monotone* orthogonal chains. Such chains are called *orthogeodesic chains* or *Manhattan* *chains* since they are shortest possible. The idea behind monotonicity is that following the course of a monotone curve is potentially easier than following the course of a curve that is allowed to make detours, an idea that is also justified by the fact that geodesic path tendency has been identified as a central graph reading behavior by Huang et al. [HEH09]. In a sense, Manhattan paths thus combine the monotonicity of straight-line drawings with the idea of a limited number of slopes from the orthogonal drawing style. Manhattan paths are geodesics with respect to the Manhattan metric. Therefore we name this drawing style the *orthogeodesic* or *Manhattan-geodesic* drawing style.

Related Work Monotone drawings have been studied from various points of view. Pach and Tóth [PT02] consider x-monotone drawings of planar graphs and show that, if each pair of edges crosses an even number of times, then there is a crossing-free x-monotone drawing, in which the x-coordinates remain unchanged and the edges are drawn as straight-line segments. Recently, Angelini et al. [ACD$^+$11] considered monotone drawings, in which each pair of vertices must be connected by a path that is monotone in some direction and presented algorithms for computing such drawings for trees and bi-connected graphs on the grid. Further, monotone drawings are related to upward-drawings [DBT88], which require edges to be directed upward, and greedy drawings [AFG09] which allow for a greedy routing scheme, in which we choose any vertex closer to the destination in every step.

In the Euclidean plane, geodesics are straight-line segments, and the classic result of König, Fáry, and Stein says that the class of graphs that have a straight-line drawing is exactly the class of planar graphs. Since there are efficient (linear-time) planarity-testing algorithms, we can decide efficiently whether a given graph has a Euclidean-geodesic drawing. We consider the same problem, which we call ORTHOGEODESIC EMBEDDABILITY, with respect to the Manhattan distance. As an example consider the graph K_4, that is, the complete graph on four vertices, which has a geodesic drawing in the Euclidean plane but not in the Manhattan plane. To avoid problems of drawing resolution, both questions are also interesting on the grid. The Euclidean case has been solved, for example, by Schnyder [Sch90] who can draw any planar n-vertex graph on a grid of size $(n-2) \times (n-2)$, which is asymptotically optimal in the worst case.

Fixed point set Next, we consider the setting, in which we are given a graph and, additionally, a finite set of points (in the plane or on the grid) to which the vertices of the graph must be mapped. We call this problem ORTHOGEODESIC POINT-SET EMBEDDABILITY. Kaufmann and Wiese [KW02] considered point-set embeddability (PSE) with respect to the *polyline drawing convention.* They showed that it is $\mathcal{NP}$-hard to decide whether a graph can be embedded on a point set with at most one bend per edge and that two bends are sufficient for any planar graph and any point set. Cabello [Cab06] showed that the problem is also NP-hard for zero bends, that is, it is NP-hard to decide whether a planar graph has a crossing-free straight-line embedding on a given point set. Everett et al. [ELLW10] showed that for every integer $n > 0$ there is a universal set U_n of n points such that every n-vertex planar graph can be 1-bend embedded on U_n.

A special case of both the straight-line and the orthogonal drawing convention has also been considered. Rappaport [Rap86] showed that it is $\mathcal{NP}$-hard to decide whether a set P of n points has an *orthogonal polygonization*, that is, whether the n-cycle can be realized on P

using horizontal or vertical edges only. O'Rourke [O'R88] proved that if one forbids 180°-degree angles in the vertices, then there exists at most one simple rectilinear polygon with vertex set P. He also showed how to reconstruct the polygon from P in $O(n \log n)$ time. We refer to Demaine's survey [Dem07] about problems related to polygonization.

Point-set embeddability with the same drawing convention but with respect to a different graph class—perfect matchings—was considered by Rendl and Woeginger [RW93]. They showed that given a set of n points in the plane, one can decide in $O(n \log n)$ optimal time whether each point can be connected to exactly one other point with an axis-parallel line segment. They also showed that the problem becomes NP-hard if one insists that the segments do not cross. Hurtado [Hur06] gave a simple $O(n \log n)$-time algorithm for the same problem under the orthogeodesic drawing convention. The idea is to alternatingly go up and down the occupied grid columns.

We will study a two-colored version of the orthogeodesic point-set embeddability problem for paths in Chapter 8. We show that it is NP-hard to decide, whether a path can be embedded on a set of n two-colored points on the grid such that no two points are horizontally or vertically aligned and such that all adjacent pairs of vertices are mapped to points with different colors. However, we also show that the problem can be solved efficiently, if there is an unoccupied column or row between each pair of points.

A somewhat related problem is that of constructing Manhattan networks. Given a set of points, find a set of axis-parallel line segments whose union contains a geodesic for each pair of points. Contrary to our setting, however, geodesics may intersect and overlap. Constructing minimum Manhattan networks, that is, networks of minimum total length, has recently been shown $\mathcal{NP}$-hard [CGS11].

Fixed correspondence We further restrict the placement of the vertices by making the bijection between vertices and points part of the input. We call the resulting problem Labeled Orthogeodesic PSE. A special case of this problem (where the graph is a perfect matching) has applications in VLSI layout. Insisting on orthogeodesic connections makes sure that signals reach their destinations as fast as possible. For example, a popular, but more restrictive wiring technique in VLSI layout, *single-bend wiring*, uses special geodesic connections with only one bend per edge. Raghavan et al. [RCS86] have shown that one can decide our perfect matching problem efficiently when insisting on at most one bend per edge.

For the same problem with given vertex–point correspondence but under the polyline drawing convention, Pach and Wagner [PW01] showed that it is possible to embed any planar graph on any set of points, but they also showed that some edges may require $\Omega(n)$ bends. For the case that one insists on at most one bend per edge, Goaoc et al. [GKO^{+}09] showed that it is $\mathcal{NP}$-hard to decide whether a given graph can be 1-bend embedded on a given set of points with given vertex–point correspondence. They also showed that it is hard to approximate the number of edges that are drawn with one bend even if one knows that the given instance is a *yes*-instance of the previous problem.

Contribution Drawing graphs in the orthogeodesic drawing style opens up a large new field of research. In this chapter we study orthogeodesic embeddability problem from an algorithmic point of view for planar graphs.

We show that Orthogeodesic Embeddability on the grid is equivalent to deciding

Table 6.1: Summary of the results for the considered orthogeodesic embedding problems; hard is short for $\mathcal{NP}$-hard.

	Orthogeodesic Embeddability	**Orthogeodesic Point-Set Embeddability**		
		unrestricted	labeled (on grid)	labeled (off grid)
Planar graph	$\mathcal{P}$ [Thm. 6.1]	hard [Th. 6.2]	hard [Th. 6.4]	$\mathcal{P}$ [Th. 6.5]
Matching	trivial	$\mathcal{P}$ [Hur06]	hard [Th. 6.4]	$\mathcal{P}$ [Th. 6.5]
Polygonization	trivial	$\mathcal{P}$ [Th. 6.3]	open	$\mathcal{P}$ [Th. 6.5]

whether the given graph has a rectilinear *one*-bend drawing on the grid in Section 6.2. Bläsius et al. [BKRW10] proposed an algorithm to decide the latter question efficiently. It is easy to see that a rectilinear one-bend drawing of an n-vertex graph fits on the $n \times n$ grid.

Further, we prove that Orthogeodesic PSE is $\mathcal{NP}$-hard on (and off) the grid, reducing (in two steps) from Hamiltonian Cycle in Section 6.3. In contrast, we give a complete and easy-to-check characterization of all *yes*-instances of Orthogeodesic Polygonization, which is the special case of Orthogeodesic PSE where the input graph is restricted to a cycle. Recall the results of Rappaport [Rap86] and O'Rourke [O'R88] concerning the corresponding problem where edges must be drawn as axis-parallel line segments, that is, without bends.

As a side note, we briefly mention that we will study orthogeodesic embeddings of trees on *general point sets*, that is, point sets with no horizontally or vertically aligned points, from a combinatorial point of view in the following Chapter 7. For different classes of trees and various additional constraints on the embedding, we present upper bounds on the minimum number of points $f(n)$ such that every tree of the given class with n vertices can be embedded according to the given constraints on every general point set with $f(n)$ points, thus, providing a sufficient characterization of a large class of point sets admitting orthogeodesic embeddings of all trees.

Finally, we show that Labeled Orthogeodesic Matching on the grid is $\mathcal{NP}$-hard by reduction from 3-Partitionin Section 6.5. This implies NP-hardness of Labeled Orthogeodesic PSE on the grid and, by a simple reduction, the hardness of finding p vertex-disjoint paths on a directed grid. The latter result complements a result of Marx [Mar04] saying that it is $\mathcal{NP}$-hard to find p *edge*-disjoint paths in grid graphs. Our proof vitally exploits the space limitation of the grid. On the other hand, we show that Labeled Orthogeodesic PSE can be solved efficiently if we loosen or drop this limitation.

We give a list of results and open questions in orthogeodesic embeddability in Table 6.1. In the remainder of the chapter, by a *grid geodesic* (or, even shorter, a *geodesic*) we mean a orthogeodesic chain connecting two grid points on the grid. A *geodesic grid embedding* (or *geodesic embedding* for short) of a graph G is a drawing of G such that the vertices of G are mapped to grid points and the edges of G are mapped to interior-disjoint grid geodesics. We only consider *4-planar graphs*, that is, planar graphs with maximum degree 4, since each point in the plane allows for at most 4 outgoing orthogonal paths. When considering an instance of Labeled Orthogeodesic PSE, we will identify the vertices of the graph and the given point set. Hence, we will not refer to the set of points as a separate input, but rather as the set of vertices of the given graph.

6.2 Orthogeodesic Embeddability

In this section we ask whether a given planar graph has an orthogeodesic embedding on the grid, that is, we allow the vertices to be mapped to arbitrary grid points. Clearly, this question makes only sense for graphs of maximum degree 4, but K_4, for instance, does *not* have a geodesic embedding on the grid.

In the following, we show that a graph admits an orthogeodesic embedding on the grid if and only if it admits an orthogonal embedding on the grid with at most one bend per edge. It is well-known, that it is $\mathcal{NP}$-hard to decide if a given graph has a rectilinear embedding on the grid without bends [GT01]. On the other hand, it is also well-known, that every 4-planar graph is 3-bend-embeddable and every planar graph is 2-bend-embeddable with the only exception of the octahedron [BK94]. Further, it is known that every series-parallel graph is 1-bend-embeddable [TNU09]. Hence, all outerplanar graphs have geodesic embeddings. Bläsius et al. [BKRW10] present an efficient algorithm for computing a 1-bend embedding of a given 4-planar graph in $\mathcal{O}(n^{2.5})$ time. Hence, we have the somewhat surprising result that we can efficiently recognize graphs that admit an orthogeodesic embedding on the grid.

Theorem 6.1. *Let $G = (V, E)$ be a planar graph. Then G has an orthogeodesic embedding on the grid if and only if G is 1-bend embeddable on the grid.*

Proof. The "if"-direction is trivially true, so we immediately turn to the "only if"-direction. Suppose that G has an orthogeodesic embedding $\mathcal{E}$ on the grid. We turn $\mathcal{E}$ into an *orthogonal representation* as introduced by Tamassia [Tam87]. Such a representation consists of lists, one for each face of the given embedding. The list for a face f has, for each edge e incident to f, an entry describing (a) the shape of e in terms of left ($-90°$) and right ($+90°$) turns, and (b) the angle that the edge makes with its successor in the cyclic order of the edges around f.

Since $\mathcal{E}$ is orthogeodesic, the angles along each edge sum up to a value in $\{-90°, 0°, +90°\}$. From the representation of $\mathcal{E}$ we compute a new representation where we replace the shape entry of each edge by the corresponding sum. The result is a valid representation since for each face the sum of the inner angles remains the same and for each vertex the sum of the angles between consecutive incident edges also remains the same. Since the new representation is valid, Tamassia's flow network [Tam87] yields a corresponding (1-bend) embedding of G. $\square$

6.3 Orthogeodesic Point-Set Embeddability

Next, we ask whether a given planar graph can be embedded on a given set of grid points. We assume that we are not given a bijection between vertices and points. We refer to this problem as ORTHOGEODESIC PSE. We prove that in general, this problem is hard, using a two-step reduction from HAMILTONIAN CYCLE. Our proof also works in the case where the orthogeodesic chains are not restricted to the grid.

We start by showing that the HAMILTONIAN CYCLE COMPLETION (HCC) problem is NP-hard by reduction from HAMILTONIAN CYCLE. HCC is defined as follows. Given a non-Hamiltonian cubic graph G, decide whether G has two vertices u and v such that $G + uv$ (i) is planar, (ii) has a Hamiltonian cycle H, and (iii) has an embedding such that u and v are incident to at most two faces on the same side of H.

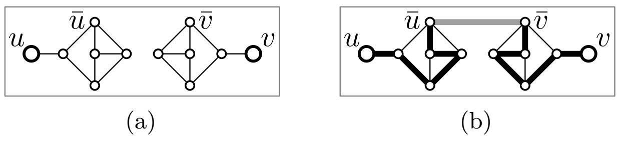


Figure 6.1: Gadget that replaces the edge uv of G in G_{uv} (when reducing from HC to HCC).

Lemma 6.1. Hamiltonian Cycle Completion *is $\mathcal{NP}$-hard.*

Proof. We reduce from the $\mathcal{NP}$-hard problem Hamiltonian Cycle (HC), where the task is to decide whether a given planar cubic graph is Hamiltonian [GJ79]. Given an instance $G = (V, E)$ of HC, we construct, for some fixed $u \in V$ and each $uv \in E$, an instance G_{uv} of HCC. The graph G_{uv} is a copy of G where we replace uv by the gadget depicted in Figure 6.1a. We claim that the edge uv lies on some Hamiltonian cycle in G if and only if G_{uv} is a *yes*-instance of HCC.

We first assume that G_{uv} is a *yes*-instance of HCC. Then there is a pair $\{a, b\}$ of vertices such that $G_{uv} + ab$ is Hamiltonian as illustrated in Figure 6.1b. The vertices a and b must lie in our gadget, one on each side (albeit not necessarily $\{a, b\} = \{\bar{u}, \bar{v}\}$); otherwise u or v would remain separators. It is obvious how to transform a Hamiltonian cycle in $G_{uv} + ab$ into a Hamiltonian cycle in G.

Conversely, assume G contains a Hamiltonian cycle H. Then H must contain u and some edge uv. We observe two things. First, if we add the edge $\bar{u}\bar{v}$ to G_{uv}, then the concatenation of $\bar{u}\bar{v}$, the bold black edges in the gadget, and $H - uv$ forms a Hamiltonian cycle $\bar{H}$ in $G_{uv} + \bar{u}\bar{v}$. Second, the planar embedding that $G_{uv} + \bar{u}\bar{v}$ inherits from G and from the embedding of the gadget as depicted in Figure 6.1b makes sure that $\bar{u}$ and $\bar{v}$ are incident to two faces on each side of $\bar{H}$.

Thus, we could apply a hypothetical algorithm for HCC to G_{uv} for each edge uv of G incident to a fixed vertex v of G. As soon as the algorithm finds a vertex pair $\{a, b\}$ such that $G_{uv} + ab$ is Hamiltonian, it is straight-forward to construct the corresponding Hamiltonian cycle in G. If, on the other hand, the algorithm decides for each edge uv of G that G_{uv} is a *no*-instance, we can conclude that G is not Hamiltonian. This yields the $\mathcal{NP}$-hardness of HCC. $\square$

Now we are ready to show the hardness of Orthogeodesic PSE.

Theorem 6.2. Orthogeodesic PSE *is $\mathcal{NP}$-hard, even for subdivisions of cubic graphs.*

Proof. Our proof is by reduction from HCC. Suppose we are given an instance $G = (V, E)$ of HCC. Note that $n = |V|$ is even, since G is a planar cubic graph and since the number of vertices with odd degree in a planar graph must be even. Let $k = n/2 + 1$. Given three non-negative integers k_0, k_1, k_2, let the point sets $P_0 = \{(-j, 0) \mid j = 0, \ldots, k_0 - 1\}$, $P_1 = \{(j, nj) \mid j = 1, \ldots, k_1\}$, $P_2 = \{(j, -nj) \mid j = 1, \ldots, k_2\}$, and $P(k_0, k_1, k_2) = P_0 \cup P_1 \cup P_2$ be constructed as illustrated in Figure 6.2a. Note that the points in $P(k_0, k_1, k_2)$ are placed such that between any two consecutive non-empty rows of the integer grid there are $n - 1$ empty rows. We now construct a graph $G' = (V', E')$ by subdividing every edge of G by a vertex of degree 2. This yields $|V'| = |V| + |E| = 2n - 1 + k$. In the following, we show

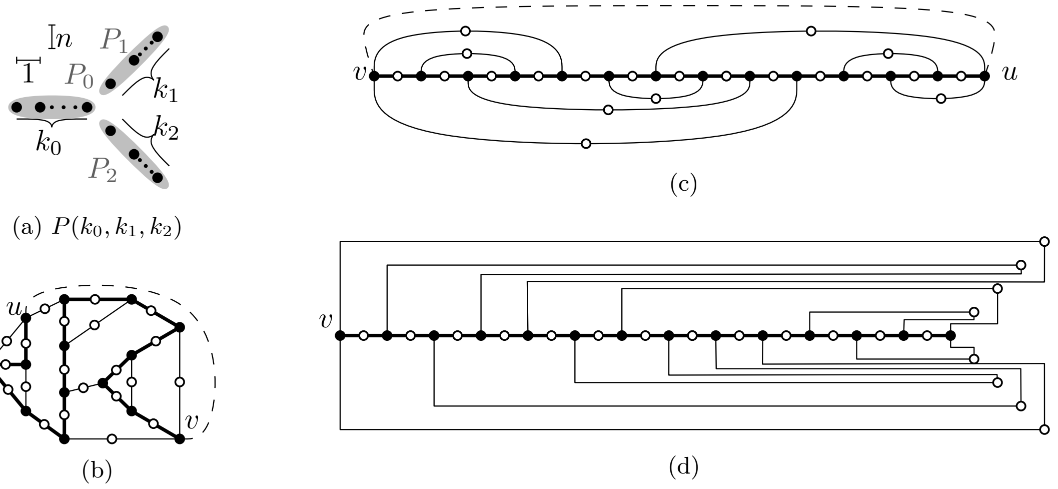


Figure 6.2: Reduction of HCC to Orthogeodesic PSE.

that G' can be embedded on $P(2n-1, k_1, k_2)$ for some k_1, k_2 with $k_1 + k_2 = k$ if and only if G is a *yes*-instance of HCC.

Assume that G is a *yes*-instance of HCC. Then there is a pair $\{u, v\}$ of vertices such that $G + uv$ contains a Hamiltonian cycle and u and v are incident to two faces on either side of this cycle. Without loss of generality, we can assume that uv is incident to the outer face. An example of a plane graph G' is depicted in Figure 6.2b; the subdivision vertices are marked by circles, the original vertices of G are marked by black disks. Maintaining the combinatorial embedding, we can embed the Hamiltonian path connecting u and v including its subdivision vertices on a set of $2n - 1$ points on a horizontal line as in Figure 6.2c. We additionally embed the faces inside the cycle above the path and the faces outside the cycle below the path. Since each vertex of G' has degree at most 3, each vertex has at most one edge going up or down—except u and v, which both have exactly one edge going up and one going down. We set k_1 and k_2 to the numbers of edges inside and outside the cycle, respectively. Then we can map the subdivision vertices of the remaining edges to the point sets P_1 and P_2, and route the edges as illustrated in Figure 6.2d. Each subdivision vertex v that is mapped to a point in $P_1 \cup P_2$ has two neighbors, a left neighbor v^- and a right neighbor v^+ (according to their x-coordinates). We route the edge vv^- with one bend and the edge vv^+ with two bends. Note that the empty rows leave enough space for all horizontal edge segments.

Conversely, assume G' has an orthogeodesic embedding on $P(2n-1, k_1, k_2)$ with $k_1+k_2 = k$. Then, the k vertices that are mapped to points in $P_1 \cup P_2$ are incident to at most $2k = n + 2$ edges. This is due to the fact that each such edge has its lexicographically larger endpoint in either P_1 or P_2, and we claim that no point in $P_1 \cup P_2$ can be adjacent to more than two lexicographically smaller points. To see the claim, note that for any point $v \in P_1$ the set of lexicographically smaller points is contained in the third quadrant with respect to v. Clearly, at most two geodesics can go from v to points in any fixed quadrant. For points in P_2, the argument is symmetric. Thus our claim holds.

Since G is cubic, G' has $3n$ edges. This leaves $3n - (n + 2) = 2n - 2$ edges incident to points in P_0 only. Since $|P_0| = 2n - 1$, P_0 induces a path π that alternates between vertices of degree 3 (original vertices) and degree 2 (subdivision vertices). There are two possibilities: either both endpoints—call them s and t—have degree 2 or both have degree 3. In the former case, π would contain $n-1$ degree-3 vertices, and s and t would be adjacent to the only remaining degree-3 vertex (not in P_0). This would mean that G is Hamiltonian—a contradiction.

Thus we may assume that s and t have degree 3. In this case, π witnesses a Hamiltonian path connecting s and t in G. This Hamiltonian path can be completed to a Hamiltonian cycle by an edge through the outer face of G. Since both u and v are incident to one edge pointing up and one edge pointing down from the path, they are incident to two faces on either side of the cycle in this embedding. This shows that G is indeed a *yes*-instance of HCC. □

Note that the proof of Theorem 6.2 implies that the result extends to the problem where the geodesics are not restricted to the grid.

6.4 Orthogeodesic Polygonization

Although Orthogeodesic PSE is NP-hard for general graphs, we can solve the problem efficiently for cycles. For a given set of grid points in the plane we wish to decide whether there is an orthogonal polygon on the grid containing all points on its boundary, such that consecutive points are connected by orthogeodesic paths. We call this problem Orthogeodesic Polygonization. We present a simple characterization of the *yes*-instances of this problem. The proof is constructive and yields an efficient algorithm that, for a given set of grid points, computes an orthogeodesic polygonization or proves that such a polygonization does not exist. Recall that the orthogonal polygonization problem is NP-hard, if the edges are not allowed to have bends [Rap86] and that, on the other hand, it can be efficiently solved if we additionally insist on right angles at the vertices [O'R88]. In contrast to this, we do allow bends on the edges, albeit only as long as the edges remain orthogeodesic chains, and we do not insist on right angles at the vertices.

In order to characterize the *yes*-instances of Orthogeodesic Polygonization, we partition the grid points in a given axis-parallel rectangle B on the grid into two groups as illustrated in Figure 6.3a. We say that a grid point p in B is *even (with respect to B)* if its rectilinear distance to the lower left corner of B is even. Otherwise, we say that p is *odd (with respect to B)*. We denote the even points by $\text{even}(P)$ and the odd points by $\text{odd}(P)$, respectively. Further, we call a set of points *degenerate* if the set is contained in an axis-parallel line. It is clear that degenerate point sets do not admit an orthogeodesic polygonization. We now characterize all point sets that *do* admit a polygonization.

Theorem 6.3. *Let P be a non-degenerate set of points on the grid, let $\mathcal{B}(P)$ be the bounding box of P, and let h and w be the numbers of rows and columns spanned by $\mathcal{B}(P)$, respectively. Then P has an orthogeodesic polygonization if and only if either (i) h or w is even or (ii) P does not contain all even points with respect to $\mathcal{B}(P)$.*

We can test, in $O(n)$ time, whether a given set of n points has a geodesic polygonization, and if so, compute one within $O(n \log n)$ time.

Note that this implies that we can always find an orthogeodesic polygonization with at least $n-1$ points by removing an arbitrary even point if P is characterized by (ii). Before

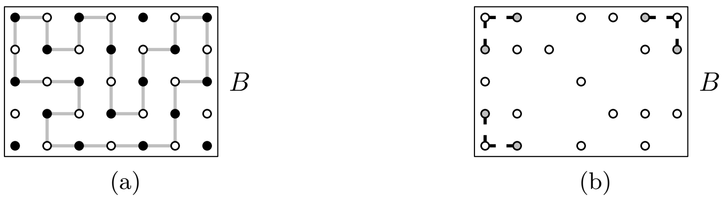


Figure 6.3: (a) A polygon hits even grid points (black disks) and odd grid points (circles) alternatingly. (b) If a point set contains a corner of its bounding box, we can assume that it also contains the (marked) points at distance 1 from that corner.

we prove this theorem, let us quickly consider the case that we are not restricted to the grid. If P is a non-degenerate set of points in the plane, we can use the grid $\Gamma(P)$ induced by P. If P fulfills the requirements of Theorem 6.3 with respect to $\Gamma(P)$, we have a very natural polygonization of P. Otherwise—if $\Gamma(P)$ has an odd number of both rows and columns, and P contains all even points with respect to $\mathcal{B}(P)$—it is sufficient to introduce *one* additional column between any two existing columns of $\Gamma(P)$ to meet the requirements of Theorem 6.3. Hence, we obtain the following corollary of Theorem 6.3.

Corollary 6.1. *Every non-degenerate set of points in the plane has an orthogeodesic polygonization off the grid. Such a polygonization can be computed in $O(n \log n)$ time.*

Proof of Theorem 6.3. Given the above characterization is correct, testing for the existence of a polygonization can clearly be done in linear time. First, we determine the bounding box, then we inspect the even points and return *yes* if either one dimension of the bounding box is even or if not all even points of the bounding box are occupied. Hence, the focus will be on the proof of the characterization. It is constructive and can be extended to an efficient algorithm. It is easy to see that each of the cases considered in the proof can be solved by a simple algorithm with running time $\mathcal{O}(n \log n)$.

Unless stated otherwise, even and odd always refers to $\mathcal{B}(P)$. We first show that P does not have a polygonization if h and w are both odd *and* P contains all even points: Observe that any polygonization of P must contain an equal number of even and odd grid points on its boundary as illustrated in Figure 6.3a. If h and w are both odd, the number of even points in $\mathcal{B}(P)$ exceeds the number of odd points in $\mathcal{B}(P)$ by one. Hence, P does not have an orthogeodesic polygonization in this case.

In the remainder of the proof, we show that we can construct a polygonization in the remaining cases. Note that the fact that P has a polygonization is invariant under rotation by multiples of 90 degrees and reflection at vertical or horizontal lines. The key idea of the proof is to partition (some rotation or reflection of) P into two sets U and $\overline{U}$ such that $\overline{U}$ contains all the points on the topmost occupied row and $U = P \setminus \overline{U}$. A *nice path* π for U is a path with the following properties.

(a) The path π connects all points in U by orthogeodesic chains.

(b) The path π ends at the topmost point of U in the leftmost column and in the topmost point of U in the rightmost column, respectively.

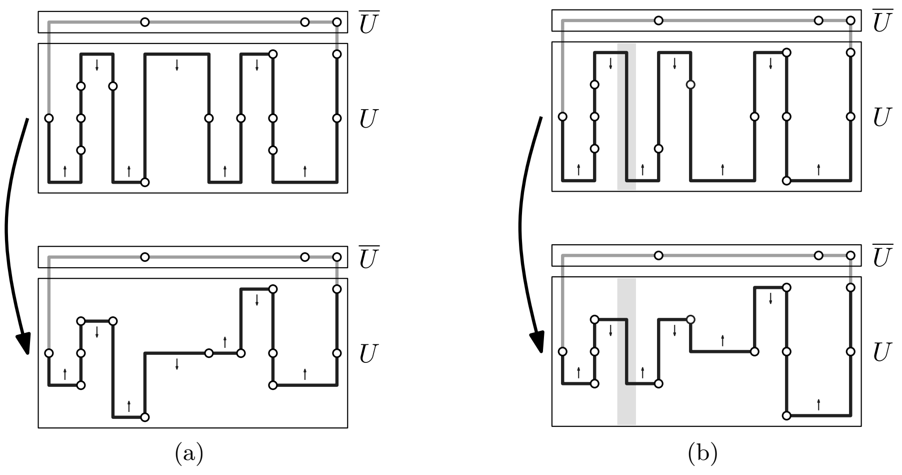


Figure 6.4: Polygonization according to Case 1: U contains (a) an even number of occupied columns or (b) an odd number of occupied columns plus (at least) one empty column, marked gray.

(c) The path π does not occupy the grid points above the two endpoints. To motivate (c), see the thick gray path in Figure 6.4a.

Note that, by definition, $\overline{U}$ is not empty, and by construction of the nice path, it is easy to connect the endpoints of the nice path in U by a path that contains all points in $\overline{U}$ such that the concatenation of the two paths yields the desired orthogeodesic polygonization of P. Thus, the problem of finding an orthogeodesic polygonization of P reduces to the problem of finding a nice path in U.

Without loss of generality, we may also assume the following. If a corner of $\mathcal{B}(P)$ lies in P, then both points in $\mathcal{B}(P)$ at distance 1 from the corner also lie in P. This follows from the fact that any polygonization containing the corner of $\mathcal{B}(P)$ must contain these two points as well as illustrated in Figure 6.3b. This observation ensures that any partition of P into $\overline{U}$ and U as described above has the property that the leftmost column and the rightmost column of U are the same as those of P, that is, any nice path can be extended to a polygonization. The proof is organized according to the case distinction illustrated in Figure 6.5. At the topmost level we distinguish between feasible and infeasible instances according to our characterization depending on the arrangement of points in B. At the next level, we distinguish two cases depending on the number of occupied columns in U. If this number is odd and all columns in U are occupied, we further partition U into sets U_{left}, U_{mid}, and U_{right}. Finally, we distinguish three cases depending on the points in U_{mid}.

Case 1 We can partition P (or some rotation or reflection of P) into $\overline{U}$ and U as described above such that either (i) the number of occupied columns in U is even or (ii) it is odd and there is at least one unoccupied column in U.

First, assume that the number of occupied columns is even. Then we can sweep the points in U from left to right and alternatingly from top to bottom and vice versa, starting at the topmost point of the leftmost column in downward direction. Imagine that all bends of our

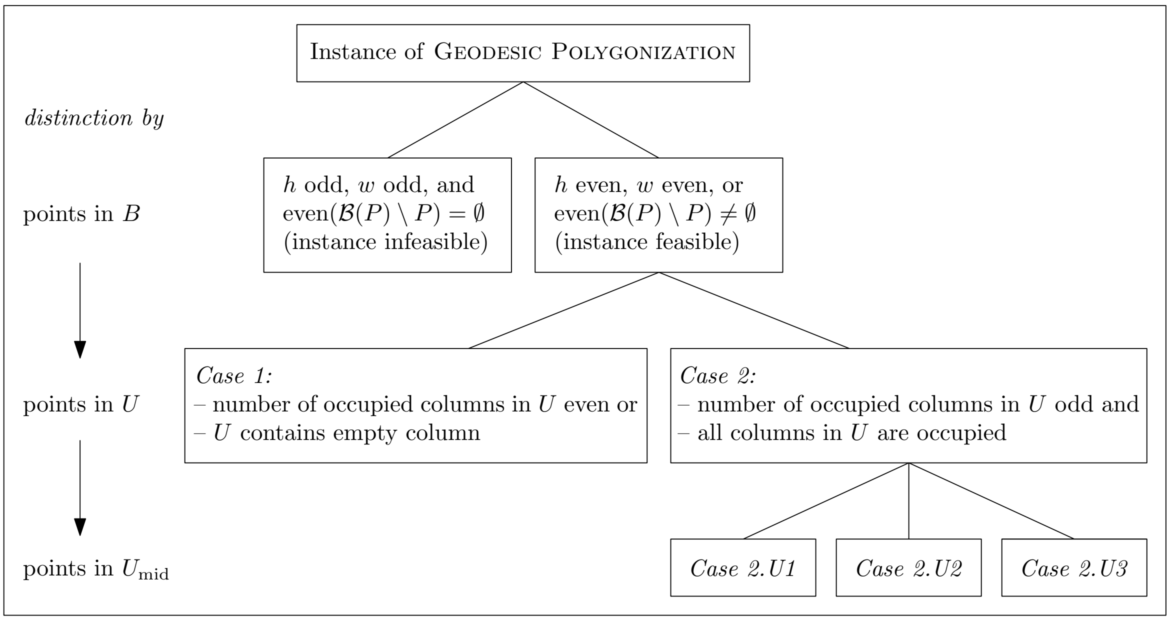

Instance of Geodesic Polygonization
distinction by
points in B
h odd, w odd, and even$(\mathcal{B}(P) \setminus P) = \emptyset$ (instance infeasible)
h even, w even, or even$(\mathcal{B}(P) \setminus P) \neq \emptyset$ (instance feasible)
points in U
Case 1: – number of occupied columns in U even or – U contains empty column
Case 2: – number of occupied columns in U odd and – all columns in U are occupied
points in U_{mid}
Case 2.U1
Case 2.U2
Case 2.U3

Figure 6.5: Case distinction in the proof of Theorem 6.3. We first distinguish between different arrangements of points in B, then in U, and finally in U_{mid}.

tour lie on the boundary of $\mathcal{B}(U)$. Then some pairs of points of U that are consecutive in our tour may be connected by U-shaped pieces of the tour, which are not geodesics. This, however, can easily be fixed by shortening each U-shape in the sense that its horizontal part is moved away from the boundary of $\mathcal{B}(U)$ until it hits at least one of the two endpoints of the U-shape. The result is either an L-shaped or simply a horizontal connection, and hence a geodesic. This process is depicted in Figure 6.4a. Since the number of columns is even, the endpoints are exactly the topmost points on the leftmost and rightmost column, respectively, and the unoccupied points above the endpoints of the path are not used.

Next, assume that the number of occupied columns is odd and there is an unoccupied column (somewhere between, but not necessarily adjacent to two occupied columns). In this case we use the same approach with the only difference that we also alternate the vertical sweeping at exactly one of the unoccupied columns. Note that this does not necessarily mean that there is a bend in the unoccupied column. As illustrated in Figure 6.4b, the last point before the unoccupied row is linked to the first point after the unoccupied row by two horizontal and one (possibly degenerate) vertical straight-line segment that uses the unoccupied column.

Case 2 We cannot partition P as described in Case 1. Then the numbers of occupied columns and rows of P are both odd, and every partition (of a rotation) of P into $\overline{U}$ and U has the property that every row and every column in U has at least one occupied point. We know that $\mathcal{B}(P) \setminus P$ contains an even point, that is, there is an unoccupied even point in $\mathcal{B}(P)$.

Before we proceed, we introduce the following notation, see Figure 6.6a. Let X be a non-degenerate set of points on the grid, and let q be a grid point in $\mathcal{B}(X)$. Then we define the

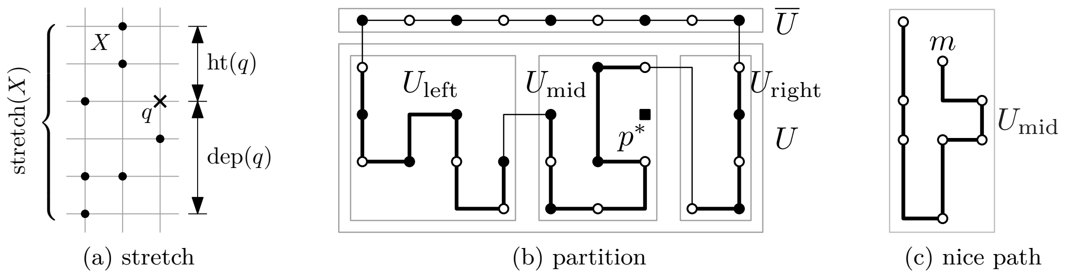


Figure 6.6: Notation for the proof of Theorem 6.3: (a) Stretch of a set X of grid points; height ht(q) and depth dep(q) of a grid point q w.r.t. $\mathcal{B}(X)$, (b) Partition of P into U and $\overline{U}$, and partition of U into U_{left}, U_{mid} and U_{right}, (c) Relaxed definition of a nice path for U_{mid} ending with m in U^2_{mid}.

height of q with respect to X to be $\max_{p \in X}\{y(p)\} - y(q)$, where $y(r)$ denotes the y-coordinate of a point $r \in \mathbb{R}^2$. Similarly, we define the *depth of q with respect to X* to be $y(q) - \min_{p \in X}\{y(p)\}$. Finally, we define the *stretch of X* to be $\max_{p \in X}\{y(p)\} - \min_{p \in X}\{y(p)\} + 1$, that is, the stretch of X is the number of rows spanned by X.

We now claim the following.

Claim 1. *In the above situation, we can find a partition of some reflection or rotation of P into two sets $\overline{U}$ and U as well as a partition of U into three sets U_{left}, U_{mid}, and U_{right} as depicted in Figure 6.6b such that the following three requirements are fulfilled.*

(R1) The set $\overline{U}$ contains the points in the topmost row of P, and $U = P \setminus \overline{U}$, as in the partition in the previous cases.

(R2) The set U_{mid} consists of the points in three consecutive columns of U with an even number of columns to the left and to the right. The corresponding subsets of U_{mid} are $U^1_{\text{mid}}, U^2_{\text{mid}}$, and U^3_{mid} (from left to right). Let $U^{23}_{\text{mid}} = U^2_{\text{mid}} \cup U^3_{\text{mid}}$, be the set of points in the rightmost two columns of U_{mid}, and let ℓ^1 denote the lowest point in U^1_{mid}. Additionally, at least one of the following three statements holds:

(U1) The stretch of $U^{23}_{\text{mid}} \cup \{\ell^1\}$ is odd.

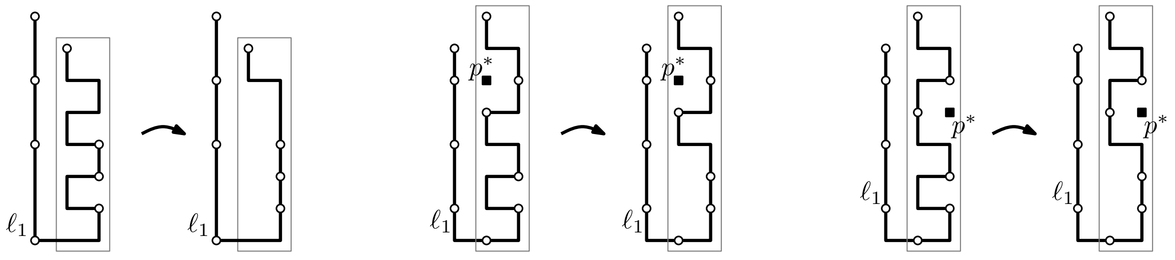
(U2) The middle column of U_{mid} contains an unoccupied point p^* with even height and odd depth with respect to $U^{23}_{\text{mid}} \cup \{\ell^1\}$.

(U3) The rightmost column of U_{mid} contains an unoccupied point p^* with odd height and even depth with respect to $U^{23}_{\text{mid}} \cup \{\ell^1\}$.

(R3) The (possibly empty) sets U_{left} and U_{right} consist of the points in the remaining columns of U to the left and to the right of U_{mid}, respectively.

We call a partition of P fulfilling (R1)–(R3) an odd *partition.*

We postpone the proof of the claim for now. Given an odd partition, we first show how to find a nice path for U by finding and linking nice paths for U_{left}, U_{mid}, and U_{right}. We slightly modify the definition of a nice path for U_{mid} by allowing the path to end in a point m of the



(a) (U1) (b) (U2) (c) (U3)

Figure 6.7: Nice paths for U_{mid} according to sub-cases (U1)–(U3). Each figure shows the sweep starting in the bottommost row of $U^{23}_{\text{mid}} \cup \{\ell_1\}$ (left) and the resulting orthogeodesic path (right).

middle column of U_{mid} if all points in U_{mid} either lie to the left or below m and if the path does not go through the point to the right of m as illustrated in Figure 6.6c.

Note that we can find nice paths for U_{left} and U_{right} as in Case 1 since both sets consist of an *even* number of occupied columns. In other words, we need only consider U_{mid}. The first part of the nice path for U_{mid} consists of a vertical straight-line segment containing all points on the leftmost column of U_{mid}. Now we follow the case distinction concerning the shape of U^{23}_{mid} in the above definition of an odd partition. Corresponding illustrations can be found in Figure 6.7.

Case 2.U1: The stretch of $U^{23}_{\text{mid}} \cup \{\ell^1\}$ is odd, as illustrated in Figure 6.7a. We sweep the second and third column of U_{mid} row by row from bottom to top, starting at the bottommost row of $U^{23}_{\text{mid}} \cup \{\ell^1\}$ and alternating the walking direction between right and left in each (not necessarily occupied) row. This yields an ordering of the points in U^{23}_{mid}. We link consecutive points in this ordering by orthogeodesic chains. These chains can be drawn such that they are interior-disjoint since, if two consecutive points have not yet been connected by a geodesic, they are either in the same column or there must be at least one empty row between them. Let m be the last point on the resulting path. Since the stretch of $U^{23}_{\text{mid}} \cup \{\ell^1\}$ is odd, the path reaches m coming from below or from the left. Hence, our path is a nice path for U^{23}_{mid}. It can be connected to ℓ_1 since the sweep goes left-to-right through the bottommost row of $U^{23}_{\text{mid}} \cup \{\ell^1\}$.

Case 2.U2: The middle column of U_{mid} contains an unoccupied point p^* with even height and odd depth with respect to $U^{23}_{\text{mid}} \cup \{\ell^1\}$, as illustrated in Figure 6.7b. In this case we compute the nice path as follows. We sweep U^{23}_{mid} from bottom to top starting at the lowest (not necessarily occupied) row in $U^{23}_{\text{mid}} \cup \{\ell^1\}$ from left to right. We alternate the walking direction between right and left in each (not necessarily occupied) row *skipping the row that contains p^**. Since the depth of p^* is odd, the walking direction is left-to-right in the row below p^*. Hence, our sweep leaves out p^*. We link points that were swept consecutively by geodesics as in the previous sub-case. Again, the resulting path is nice.

Case 2.U3: The rightmost column of U_{mid} contains an unoccupied point p^* with odd height and even depth with respect to $U^{23}_{\text{mid}} \cup \{\ell^1\}$, as illustrated in Figure 6.7c. This case

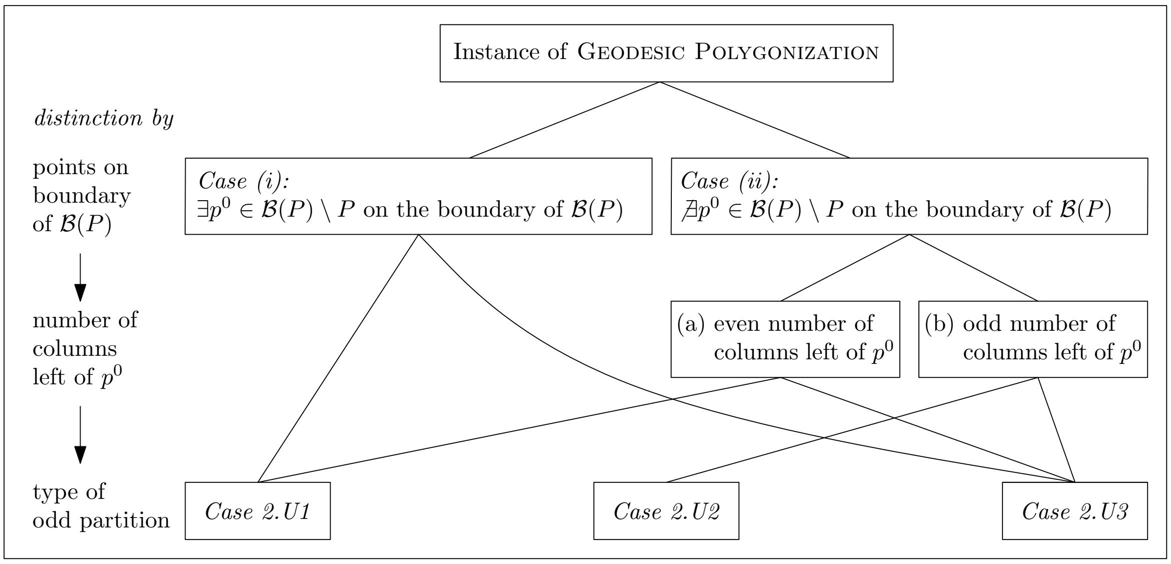

Instance of Geodesic Polygonization
distinction by
points on boundary of $\mathcal{B}(P)$
number of columns left of p^0
type of odd partition
Case (i): $\exists p^0 \in \mathcal{B}(P) \setminus P$ on the boundary of $\mathcal{B}(P)$
Case (ii): $\nexists p^0 \in \mathcal{B}(P) \setminus P$ on the boundary of $\mathcal{B}(P)$
(a) even number of columns left of p^0
(b) odd number of columns left of p^0
Case 2.U1
Case 2.U2
Case 2.U3

Figure 6.8: Case distinction for the proof of Claim 1 regarding the construction of an odd partition. Depending on the existence of a an unoccupied point on the boundary of $\mathcal{B}(P)$ we further distinguish between an even and an odd number of points to the left of the unoccupied point.

is very similar to the previous sub-case. We skip the row containing p^* in the sweep. Since the depth of p^* is even, the walking direction in the row below p^* is right-to-left. Hence, we leave out a point in the right column, which is exactly p^*.

In all three cases the two geodesic paths for U^1_{mid} and U^{23}_{mid} can be combined to a nice path for U_{mid} since the path for U^{23}_{mid} (a) starts at the leftmost point on the lowest row of $U^{23}_{\text{mid}} \cup \{\ell^1\}$ and (b) stops at the topmost point of U^{23}_{mid} without going through the point to the right of the topmost point. Hence, we have found a nice path for U_{mid}. We now turn to the claim whose proof we postponed before.

Proof of Claim 1 Suppose that Case 2 applies, that is, the numbers of occupied columns and rows of P are both odd and the partition of any rotation of P into $\overline{U}$ and U has the property that every column in U has at least one occupied point and there is an even point in $\mathcal{B}(P) \setminus P$.

In this setting, we show how to explicitly construct an odd partition of P. Note that any rotation and any reflection of the point set maps even points to even points and odd points to odd points with respect to the new positions, respectively. In order to prove the claim, we consider two cases as illustrated in Figure 6.8:

Case (i): There is an unoccupied even point p^0 on the boundary of $\mathcal{B}(P)$, as illustrated in Figure 6.9a. We assume without loss of generality that p^0 is in the bottom row r_0 of $\mathcal{B}(P)$. Since r_0 is non-empty, there must be an occupied point on r_0, say, to the left of p^0. Let p^* be the leftmost unoccupied even point of r_0 such that there is an occupied point to the left of p^*. Since p^* is an even point in r_0, there is an even number

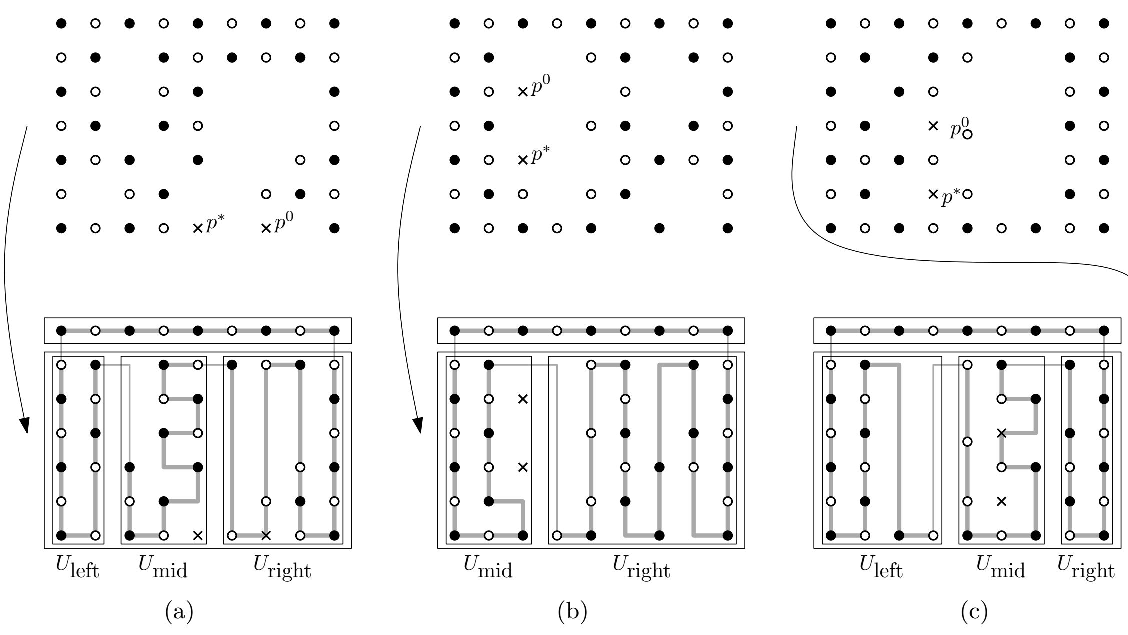


Figure 6.9: Case distinction for proving the existence of an odd partition.

of occupied columns in U to the left of p^*. By the choice of p^* this number of columns is at least 1 and hence there are at least two occupied columns to the left of p^*.

Let $\overline{U}$ be the set of points in the top row of P. Let $U = P \setminus \overline{U}$, and let U_{mid} be the subset of U in the column of p^* and the two columns to its left. Since p^* is an even point on the lowest row, there must be an even number of columns to the left and right of U_{mid}, respectively, and we fix U_{left} and U_{right} accordingly.

Since p^* is on the lowest row the depth of p^* with respect to $U_{\mathrm{mid}}^{23} \cup \{\ell^1\}$ is zero and, hence, even. If the stretch of $U_{\mathrm{mid}}^{23} \cup \{\ell^1\}$ is even, then the height of p^* with respect to U_{mid}^{23} is odd and its depth is even. Since p^* is in the rightmost column with respect to U_{mid}, this yields an odd partition according to Case (U3). If the stretch of $U_{\mathrm{mid}}^{23} \cup \{\ell^1\}$ is odd, this yields an odd partition according to Case (U1).

Case (ii): All even points on the boundary of $\mathcal{B}(P)$ are occupied, as illustrated in Figures 6.9b and 6.9c. Again, we let $\overline{U}$ be the top row of $\mathcal{B}(P)$ and let $U = P \setminus \overline{U}$. Let p^0 be a leftmost unoccupied even point in $\mathcal{B}(U)$. Such a point must exist, since all even points on the boundary are in P and $\overline{U}$ only contains points on the boundary. This also implies that there is at least one occupied column to the left of p^0. Note that by assumption all even points to the left of p^0 are occupied. We distinguish two cases:

(a) First, suppose that there is an even number of columns to the left of p^0 as illustrated in Figure 6.9b. Let U_{mid} consist of the points in the column of p^0 and the two columns to its left and choose U_{left} and U_{right} accordingly. Let p^* be the lowest unoccupied even point in the column of p^0. Since p^* is an even point with an even number of columns to the left and since all even points below p^* are occupied, p^* has an even depth with respect to U_{mid}^{23}. If the height of p^* is even, then the stretch of U_{mid} is odd, hence, this yields an odd partition according to Case (U1). Otherwise, this yields an odd partition according to Case (U3).

(b) Next, assume that there is an odd number of columns to the left of p^0 as illustrated in Figure 6.9c. In this case we mirror the instance on a vertical line, that is, p^0 is in the rightmost column containing an unoccupied even point and there is an odd number of columns to the right of p^0 whose even points are all occupied. We let U_{mid} consist of all the points of U in the columns of p^0 and its two neighboring columns and choose U_{left} and U_{right} accordingly. Let p^* be the lowest unoccupied even point on the column of p^0. Since p^* is an even point with an odd number of columns to the left and since all even points on the column to its right are occupied, p^* has an odd depth with respect to U_{mid}. If the height of p^* is odd as well, then the stretch of U_{mid} is odd, and therefore, this yields an odd partition according to Case (U1). Otherwise the height of p^* is even, and this yields an odd partition according to Case (U2).

This finishes the proof of our claim and, thus, we can always find an odd partition in Case 2 that can be used to compute an orthogeodesic polygonization. Hence, the theorem holds. □

6.5 Labeled Orthogeodesic Point-Set Embeddability

Since the orthogeodesic point-set embedding problem is NP-hard both on and off the grid, we next investigate the complexity of Labeled Orthogeodesic PSE, a variant of the orthogeodesic embedding problem in which we are given a *geometric* graph G as an input whose vertices are associated with a given set of points on the grid. Since the position of the vertices of G are given, the problem is that of finding an orthogeodesic embedding of the edges of G. We show that this problem is $\mathcal{NP}$-hard, even for perfect matchings. Interestingly, however, it turns out that the problem can be solved efficiently if we drop the restriction imposed by the grid. Note that this behavior is in contrast to that of Orthogeodesic Point-Set Embeddability, where the vertex–point correspondence is not given. We showed that this problem $\mathcal{NP}$-hard on *and* off the grid in Section 6.3.

Theorem 6.4. Labeled Orthogeodesic PSE *on the grid is $\mathcal{NP}$-hard, even if the given graph is a perfect matching.*

Proof. To prove the theorem we reduce 3-Partition to Labeled Orthogeodesic Matching (LGM), which is a special case of Labeled Orthogeodesic PSE in which the graph is a perfect matching. An instance of 3-Partition consists of a multiset $A = \{a_1, \dots, a_{3m}\}$ of $3m$ positive integers, each in the range $(B/4, B/2)$, such that $B = (\sum_{a \in A} a)/m$, and the question is whether there exists a partition of A into m subsets $A_1, \dots, A_m$ of A, each of cardinality three, such that the sum of the numbers in each subset is B. Since 3-Partition is *strongly* $\mathcal{NP}$-hard [GJ79], we may assume that B is bounded by a polynomial in m.

Based on an instance A of 3-Partition, we now construct an instance M of LGM consisting of pairs of grid points that must be connected by an orthogeodesic chain such that M is a *yes*-instance of LGM if and only if A is a *yes*-instance of 3-Partition. Figure 6.10 shows an example instance M for a multiset A of nine numbers. The instance M consists of three types of point pairs.

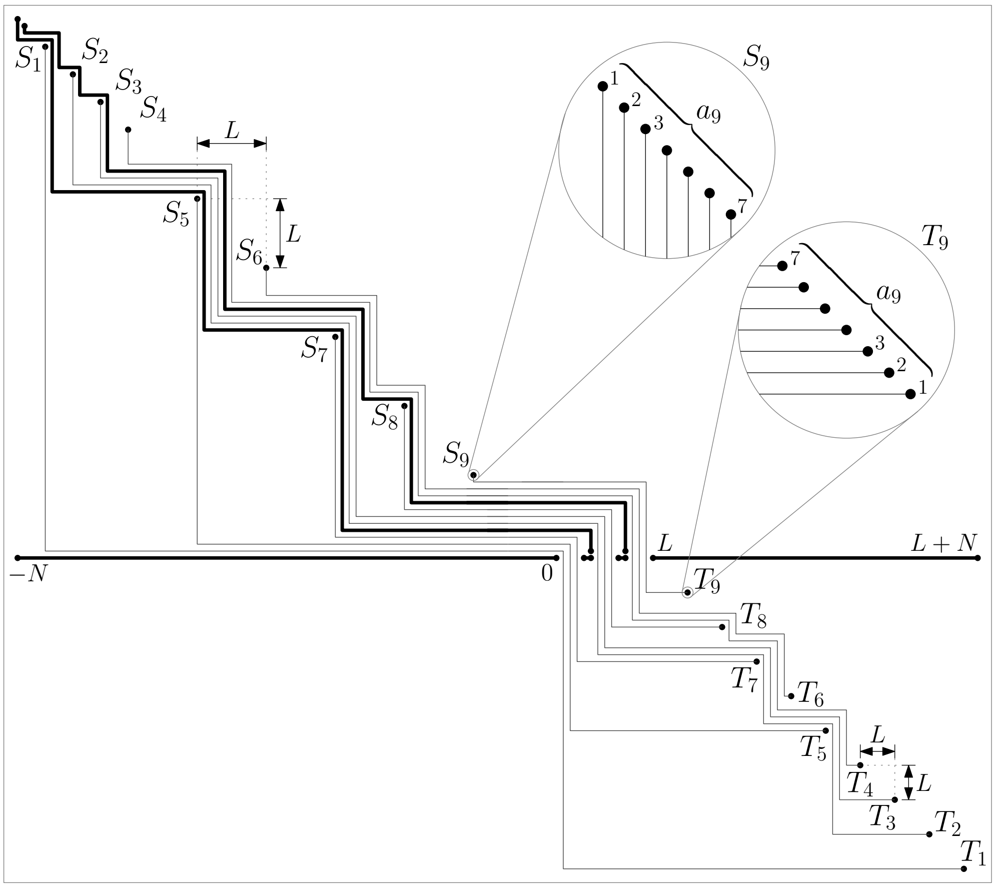


Figure 6.10: Example of the reduction from 3-Partition to LGM using $A_1 = \{a_1, a_5, a_7\}$, $A_2 = \{a_2, a_3, a_8\}$, and $A_3 = \{a_4, a_6, a_9\}$ (not to scale).

The first type of point pairs represents the numbers in A and consists of $6m$ sets of points $S_1, \ldots, S_{3m}$ and $T_1, \ldots, T_{3m}$ such that the points in S_i are associated with the points in T_i as we will explain below.

The $3m$ sets $S_1, \ldots, S_{3m}$ of grid points, all lie on the diagonal $\ell : y = -x$, in this order from left to right. For $1 \leq i \leq 3m$, the points in S_i occupy a_i consecutive grid points, and two consecutive sets are separated by a large gap of $L = Bm + m - 1$ grid points. The gap between the last point of S_{3m} and the origin is also L.

The points in the sets $T_{3m}, \ldots, T_1$ lie on the line $\ell' : y = -x + L$, in this order from left to right. Again, points within a set are consecutive grid points, and between consecutive sets there are large gaps of L grid points. The matching is as follows. For $1 \leq i \leq 3m$ and for $1 \leq j \leq a_i$, the j-th point in S_i (counting from the *left*) matches the j-th point in T_i (counting from the *right*). The a_i point pairs in $S_i \cup T_i$ represent the number a_i.

The second type of point pairs forms a sort of “dot mask” forming the partitions. These pairs lie on the x-axis. Therefore, the geodesics between them are obviously line segments and pairwise disjoint. The leftmost segment matches $-N$ to 0, where $N = 3mL + mB + 2(m-1)$. The following $m - 1$ segments have unit length and leave gaps of width B. The rightmost segment matches L to $L + N$.

Finally, the third type of point pairs gives rise to geodesics that will act as partitioners ensuring that all geodesics that represent a number from A go through the same gap in the mask. There are $m - 1$ such pairs. Their upper endpoints are consecutive grid points on the diagonal ℓ. They lie above the points in S_1, leaving a gap of $m - 1$ grid points. The corresponding lower endpoints lie one unit above the right endpoints of the unit-length segments on the x-axis forming the dot mask. The matching is such that, from left to right and for $1 \leq j \leq m-1$, the j-th upper endpoint matches the j-th lower endpoint, as illustrated in Figure 6.10.

It is easy to see that any orthogeodesic embedding of M induces a partition of A. Since the partitioners cannot pass between points associated with the same number in A and since there is only one gap in the dot mask between two partitioners, all edges corresponding to the same element of A must be routed through the same gap of the dot mask, each of the m gaps has width B, and each of the mB edges must go through some gap.

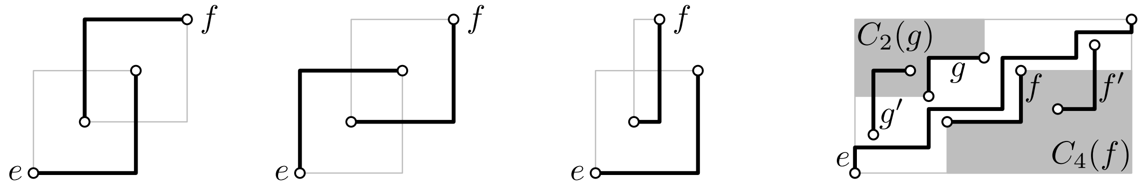
Conversely, given a partition, we can construct an orthogeodesic embedding of the matching as follows. We start by drawing the dot mask whose layout only depends on the numbers B and m and which is uniquely determined by the placement of the respective endpoints. Then, we analyze the first subset of the partition, A_1, and connect the points in $S^1 = \bigcup_{a_j \in A_1} S_j$ to the corresponding points in $T^1 = \bigcup_{a_j \in A_1} T_j$, starting with the leftmost point in S^1 and the rightmost point in T^1. For each connection, we use the bottommost orthogeodesic chain that is routed one grid unit above and to the right of all orthogeodesic chains we have drawn so far. Next, we draw the first (that is, leftmost) partitioner. Also in this case, we use the bottommost orthogeodesic chain that is routed above all orthogeodesic chains we have drawn so far. We repeat these two steps, connecting the points corresponding to a subset of the partition and drawing a fence. Since we left enough horizontal and vertical space between the points representing the numbers in A, this process does not create any crossings among the constructed orthogeodesic chains.

Since we assumed that B is polynomial in m, the numbers L and N, which determine the grid size needed by M, are also polynomial in m. Thus, given an embedding, the partition can be constructed from this embedding efficiently, and vice versa. Thus our reduction is polynomial, which concludes the proof. □

6.6 Sparse Labeled Orthogeodesic Point-Set Embeddability

Contrary to the previous section, we show that Labeled Orthogeodesic PSE becomes easy if we loosen or drop the space limitation of the grid. We call an instance $G = (V, E)$ of Labeled Orthogeodesic PSE *sparse* if the minimum distance between any two occupied columns and between any two occupied rows is at least $3n-5$. In the remainder of this section, we give an efficient algorithm that solves sparse instances of Labeled Orthogeodesic PSE. Clearly, the algorithm can also be used for an instance that does not "live" on the grid, by underlaying the instance with a fine enough grid. Throughout the section we will identify vertices and points according to the given mapping and we will assume that the graph does not contain isolated vertices. Isolated vertices can be treated as single vertices with a loop that is not embedded.

First, we derive the notion of a combinatorial (geodesic) embedding of a geometric graph, that is, a graph whose vertices are mapped to points in the plane, as well as a set of necessary



(a) e below f (b) f below e (c) e strictly below f (d) $f \prec e \prec g$, $g' \in C_2(g)$, $f' \in C_4(f)$

Figure 6.11: Combinatorial orthogeodesic embedding of the edges of a geodesic graph.

conditions for the existence of such an embedding. Then we show that we can efficiently compute an orthogeodesic embedding from a combinatorial embedding. Finally, we show how to efficiently obtain a combinatorial embedding or provide a short and comprehensible proof that no such embedding exists.

We say that an edge $e \in E$ is *downward* if its lexicographically larger endpoint e^+ lies below its lexicographically smaller endpoint e^-, otherwise e is *upward*. We consider horizontal or vertical edges to be both upward and downward, respectively.

A vertex v of a 4-planar geometric graph, that is a graph with given vertex positions, is called *admissible* if it is incident to at most one vertex on each of the rays starting in v, to at most two vertices in each (closed) quadrant with respect to v and at most three vertices in each (closed) axis-aligned half-plane with respect to v. Clearly, a 4-planar geometric graph can only admit an orthogeodesic embedding if it is admissible, that is, if all its vertices are admissible.

Let γ be an orthogeodesic embedding of G. We say that $\gamma(e)$ is *below* $\gamma(f)$ (and that $\gamma(f)$ is *above* $\gamma(e)$) if there is a vertical line ℓ intersecting $\gamma(e)$ below $\gamma(f)$ as illustrated in Figure 6.11a. Note that ℓ intersects the bounding boxes of both e and f in this case. Our goal is to provide a combinatorial description of the possible embeddings for the edges of G in terms of these above–below relations. In other words, for each pair (e, f) of edges such that there is a vertical line intersecting the bounding boxes of both e and f, we would like to specify whether e is below or above f. Hence, we are looking for a partial order on the set of edges as a combinatorial description of the embeddings. Given an orthogeodesic embedding of G, it is easy to derive such a partial order. Not every partial order on the edges, however, corresponds to an orthogeodesic embedding of G. We say that an edge e is *strictly below* an edge f (and that f is *strictly above* e) with respect to G if and only $\gamma(e)$ is below $\gamma(f)$ in *every* orthogeodesic embedding of G as illustrated in Figure 6.11c.

We denote the set of points in the (closed) k-th quadrant of the coordinate system centered at $p \in \mathbb{R}^2$ by $Q_k(p)$. We say that $p \in \mathbb{R}^2$ *k-dominates* $q \in \mathbb{R}^2$ if $q \in Q_k(p)$. Similarly, for $e \in E$, let $Q_k(e) := Q_k(e^-) \cup Q_k(e^+)$; we say that e k-dominates $q \in \mathbb{R}^2$ if $q \in Q_k(e)$. In Figure 6.11d, for example, the edge g 2-dominates $(g')^-$ and f 4-dominates $(f')^-$.

We define the *k-critical set of edges of* e as $C_k(e) := \{f \in E \mid f \cap Q_k(e) \neq \emptyset\}$, that is, the set of edges with at least one endpoint in the region that is k-dominated by the endpoints of e. Consider an upward edge e that is strictly above an edge f and let $f' \in C_4(f)$. This implies that e is strictly above f' as well, since any geodesic upward chain that passes above some point p must also pass above all points that are 4-dominated by p as illustrated in

Figure 6.11d. Similar observations can be made for the case that e is strictly below f as well as for downward edges. Note that the following dualities hold for all edges e and f

$$f \in C_4(e) \Leftrightarrow e \in C_2(f) \text{ and } f \in C_1(e) \Leftrightarrow e \in C_3(f).$$

Clearly, an upward edge e is strictly above all edges in $C_4(e)$ and strictly below all edges in $C_2(e)$. Similarly, a downward edge is strictly above all edges in $C_3(e)$ and strictly below all edges in $C_1(e)$. For instance, in Figure 6.11c, edge f is in $C_2(e)$ and, therefore, e is strictly below f. By incorporating these observations, we obtain the following notion of a combinatorial orthogeodesic embedding.

A *combinatorial orthogeodesic embedding of G* is a partial order $\prec$ on the set of edges E such that two edges are comparable whenever there is a vertical line intersecting the bounding boxes of both edges and such that the following *implication rules* hold.

Implication Rules. *Let $e \in E$ be an upward edge, and let $f \in E$. If $f \in C_2(e)$, then $e \prec f$. If $f \in C_4(e)$, then $f \prec e$. Further, if $f \prec e$, then $f' \prec e$ for all $f' \in C_4(f)$. If $e \prec f$, then $e \prec f'$ for all $f' \in C_2(f)$.*

Similarly, let $e \in E$ be a downward edge and let $f \in E$. If $f \in C_1^e(e)$, then $e \prec f$. If $f \in C_3^e(e)$, then $f \prec e$. Further, if $f \prec e$, then $f' \prec e$ for all $f' \in C_3^e(f)$. If $e \prec f$, then $e \prec f'$ for all $f' \in C_1^e(f)$.

These implication rules summarize our observations from above. Whenever one of the rules is violated in a given partial order $\prec$ for any edge e, this implies that the corresponding order cannot be realized by a geodesic chain, which implies that the partial order does not correspond to any partial order that can be obtained from an orthogeodesic embedding of G. Now that we have an understanding of a combinatorial orthogeodesic embedding, we can show that the notion is well-defined by showing that every combinatorial embedding of G corresponds to an orthogeodesic embedding of G. Given a combinatorial embedding $\prec$, let $G_\prec = (E, \mathcal{A})$ denote the corresponding directed graph on the set of edges E with an edge $(e, f) \in \mathcal{A}$ if and only if $e \prec f$.

Lemma 6.2. *Given a combinatorial orthogeodesic embedding $\prec$ of an admissible geometric graph $G = (V, E)$ represented by $G_\prec$, we can compute a geodesic embedding of G according to $\prec$ in $\mathcal{O}(n^2)$ time, which is worst-case optimal.*

Proof. In order to compute an orthogeodesic embedding according to $\prec$, we sweep a vertical line over the point set. Events occur at the vertices of G, sorted in lexicographical order from left to right and from bottom to top. During the sweep, we partition the edges in E into three groups. *Completed edges* have both endpoints to the left of the sweep-line. We have already embedded these edges as orthogeodesic chains. *Partial edges* have one endpoint on either side of the sweep-line. A partial edge is embedded as a *partial geodesic* ending at the sweep-line. Finally, *untouched edges* have both endpoints to the right of the sweep-line. We have not started embedding these edges yet. During the sweep we maintain the following invariants.

1. All completed and partial edges are (partially) embedded as geodesics.

2. For every partial downward edge the partial embedding is not upward; vice versa for partial upward edges.

3. The ordering of the endpoints of the partial edges from bottom to top corresponds to the partial order $\prec$.

4. No two (partial) geodesics intersect.

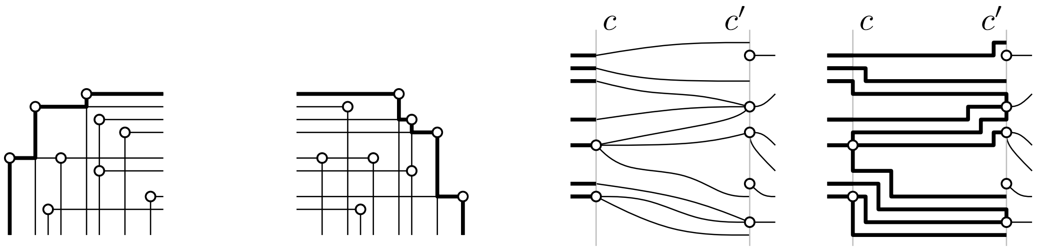
Let c and c' be two consecutive occupied grid columns, with c to the left of c'. Assume that we have already computed a partial geodesic embedding up to c. Essentially, we extend the edges not ending at c in an iterative fashion from bottom to top such that each newly embedded chain is just above all previously embedded chains, as well as above the endpoints on c' that correspond to edges below the current edge. To this end, we maintain a sorted list of the edges at the sweepline. In more detail, we proceed as follows.

First, we sort the events in lexicographical order in $\mathcal{O}(n \log n)$ time. Let the resulting order be given by $v_1, \ldots, v_n$. Next, we sort the edges topologically according to $\prec$, resulting in a total order on the edges, say $e_1, \ldots, e_m$. Given $G_\prec$, we can sort the edges topologically in time proportional to the number of edges of $G_\prec$, which is in $\mathcal{O}(n^2)$, since $G_\prec$ is a simple graph whose vertices correspond to the edges of a planar graph. Note that, whenever the sweepline intersects the bounding boxes of a set of edges, these edges must be totally ordered by $\prec$ and therefore, the sequence of partial edges at the sweepline, sorted from bottom to top, must be a subsequence of $e_1, \ldots, e_m$. In order to compute the embedding, it essentially suffices to merge the order of the events with the order of the edges. This way, we can compute an embedding in $\mathcal{O}(n^2)$ time.

To merge the order defined by the events with the order defined by the edges, we proceed as follows. At the sweepline we store a list of partial edges sorted according to $\prec$. Let $e_{\alpha(1)}, \ldots, e_{\alpha(k)}$ be the set of partial edges that are stored at the sweepline when it is located at c, that is, $\alpha(1) < \cdots < \alpha(k)$. Further, let v be the next event on c'. Suppose, we would like to embed $e_{\alpha(i)}$ for some i. Hence, we need to decide whether $e_{\alpha(i)}$ must be embedded above or below v. Since there are only $\mathcal{O}(n)$ vertices and edges, respectively, we can store this information in $\mathcal{O}(n^2)$ space and access it in $\mathcal{O}(1)$ time. Whenever $e_{\alpha(i)}$ is the first edge on c that is above v, we insert the edges starting in v into the sequence of edges stored at the sweepline before embedding $e_{\alpha(i)}$, such that it will be considered in the subsequent steps of the algorithm.

The merging can only be done if the following holds. Assume that $e := e_{\alpha(i)}$ is an upward edge that must be embedded above v according to $\prec$ and recall that we did not allow isolated vertices for technical reasons, that is, v is the endpoint of some edge, say e'. Then (1) none of the following edges $e_{\alpha(j)}$ with $j > i$ may be embedded below v and (2) e must be embedded above all edges in $C_4(e')$ and below all edges in $C_2(e')$. First, assume for contradiction that $f := e_{\alpha(j)}$ is an edge with $j > i$ that must be embedded below v. Then we have $e' \prec e$ since e must be embedded above v and $f \prec e'$ since f must be embedded below v. However, this implies $f \prec e$ by transitivity of $\prec$. On the other hand, since both $\mathcal{B}(e)$ and $\mathcal{B}(f)$ are intersected by the sweepline at c, they are comparable with respect to $\prec$ and, therefore, $\alpha(i) < \alpha(j)$ implies $e \prec f$, a contradiction to the acyclicity of $\prec$. Second, since $\prec$ respects the implication rules, (2) is trivially fulfilled by $\prec$. The case, when e is a downward edge can be handled similarly.

As another ingredient, we describe how to embed single edges. Note that, an upward (downward) Manhattan-geodesic chain is uniquely determined by the 4-dominant (3-dominant) points corresponding to the right (left) bends along the chain. We will use this observation to describe the orthogeodesic chains corresponding to single edges. Let R be a set of points



(a) 4-hull (b) 3-hull (c) extension of a partial embedding

Figure 6.12: Computing an orthogeodesic embedding from a combinatorial embedding.

and let e be an upward edge. By $R^{\nwarrow} := \{(x-1, y+1) \mid (x,y) \in R\}$ we denote the set of points resulting from translating R one unit of the grid to the left and to the top, respectively. Similarly, by $R^{\nearrow} := \{(x+1, y+1) \mid (x,y) \in R\}$ we denote the set of points resulting from translating R one grid unit to the right and one grid unit to the top. Further, we define the *k-hull* of R as the boundary of the regions $\bigcup_{p \in R} Q_k(p)$, denoted by $H_k(R)$, that is, $H_k(R)$ is the boundary of the set of points that are k-dominated by R. See Figure 6.12 for an illustration. Hence, in order to compute a orthogeodesic chain for a new edge in the sequence computed above, we simply compute the k-hull of the set of points corresponding to bends of the previously embedded edges and translate this k-hull one unit to the top and either to the left or to the right, depending on the type of edge. Let B_i be the set of endpoints of all straight-line segments used between c and c' in the embedding up to this point and consider the edge $e = e_{\alpha(i)}$. Let $p_{\alpha(i)}$ be the tentative endpoint of $e_{\alpha(i)}$ on c and let $v \in V$ be the vertex corresponding to the current event. If $e_{\alpha(i)}$ is an upward edge that does not end on c', then we embed it on the fraction of the 4-hull of the point set $(B_i \cup \{v\} \setminus e_{\alpha(i)})^{\nwarrow} \cup \{p_{\alpha(i)}, e^+_{\alpha(i)}\}$ between c and c', that is, it is embedded just above all (partial) geodesics of edges $e_{\alpha(j)}$ with $j < i$. Similarly, if $e_{\alpha(i)}$ is a downward edge that does not end on c', we embed it on the fraction of the 3-hull of the point set $(B_i \cup \{v\} \setminus e_{\alpha(i)})^{\nearrow} \cup e_{\alpha(i)} \cup e_a$ between c and c'. If $e = e_{\alpha(i)}$ is an upward edge ending on c', we embed it as follows. If e^+ is the left endpoint of two downward edges or one downward edge and two upward edges or if e is the second of two upward edges ending in e^+, we embed e on the fraction of the 4-hull of the point set $(B_i \setminus e_{\alpha(i)})^{\nwarrow} \cup \{p_{\alpha(i)}, e^+, q_e\}$ between c and c', where q_e is the gridpoint to the left of e^+. Otherwise, we embed e on the fraction of the 4-hull of the point set $(B_i \setminus e_{\alpha(i)})^{\nwarrow} \cup \{p_{\alpha(i)}, e^+\}$ between c and c'. This special treatment is necessary to reserve the space for the edges starting at e^+ to the right. Downward edges ending on c' are treated similarly. See Figure 6.12 for an illustration.

From a practical point of view, it is more convenient to use the k-hull of the previous step in order to compute the k-hull for the current edge instead of using the endpoints of all edges embedded so far. In this case, the new k-hull can be computed in linear time by walking along the given k-hull. The total time complexity for this is proportional to the number of bends in the resulting embedding. Note, that each bend can be attributed to a vertex of the graph. Hence, each edge has at most n bends, resulting in a total of at most n^2 bends. The total time complexity of the algorithm is, therefore, bounded by $\mathcal{O}(n^2)$.

It remains to show that the algorithm is correct, that is, it maintains the embedding and

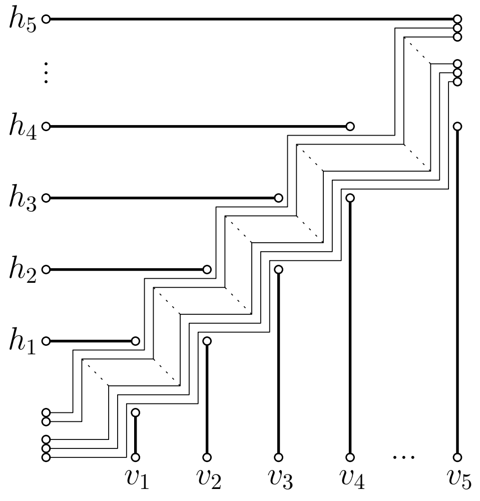


Figure 6.13: Instance of Labeled Orthogeodesic Matching with $3n$ edges and at least $2n^2$ bends.

does not create any crossings between the edges. Clearly, the embedding is preserved if the instance is sparse and, thus, contains at least $3n - 6$ unoccupied columns and rows between each pair of occupied columns and rows, respectively. To see this, recall that, in order to compute the embedding for an edge, we merely translated the k-hull of the set of previously computed bends by one unit to the top and to the left or right, respectively. Since a planar graph contains at most $3n - 6$ edges, we never run out of space this way, implying that the combinatorial embedding is maintained throughout the algorithm. Since the edges are always embedded above all endpoints of previously embedded straight-line segments, the resulting embedding must also be crossing-free. Additionally, no segment of the grid incident to the vertices of the graph is used by more than one edge, since the graph is admissible.

In order to see that the algorithm is worst-case optimal, consider the graph illustrated in Figure 6.13. It consists of n horizontal edges $h_1, \dots h_n$ and n vertical edges $v_1, \dots, v_n$ as well as n upward edges. Clearly, each of the upward edges is strictly above the vertical edges and strictly below the horizontal edges. Due to the arrangement of the horizontal and vertical edges, each pair of edges (h_i, v_i) causes each of the upward edges to bend, resulting in $\Omega(n^2)$ bends. □

The previous lemma shows that a combinatorial description of the embedding can be turned into an orthogeodesic embedding in $\mathcal{O}(n^2)$ time. Clearly, we can modify the algorithm such that it can be used to test, whether a given combinatorial embedding has an orthogeodesic embedding on the grid. To this end, we simply stop whenever there is not sufficient space. Hence, Labeled Orthogeodesic PSE is fixed-parameter tractable with respect to the number of combinatorial embeddings.

Next, we show how to compute a combinatorial embedding efficiently. We show how to compute a directed graph $\Pi = (E, A)$ on the set of edges of G, called a *pre-embedding of* G, which contains only edges (e, f) such that e is strictly below f and whose acyclicity is equivalent to the existence of a combinatorial embedding. Moreover, this graph in a sense implicitly encodes all combinatorial embeddings.

During the computation, we essentially compute the fix-point obtained by repeatedly applying the implication rules to an initial partial order $\prec_0$ of strictly-below relations. This yields a sequence $\prec_0, \prec_1, \ldots$ of partial orders. As long as $\prec_i \neq \prec_{i+1}$ we iteratively derive more necessary constraints for the embedding. We stop this process as soon as $\prec_i = \prec_{i+1}$. Geometrically, this corresponds to computing lower and upper orthogeodesic boundaries for each edge e, defined by the union of the boundaries of the regions dominated by the elements that are strictly below and above e, respectively.

Lemma 6.3. *A pre-embedding* Π *of* G *can be computed in* $\mathcal{O}(n^2 \log n)$ *time. If the pre-embedding is acyclic, we can compute a combinatorial embedding within the same time.*

Proof. We compute Π as follows. For each edge e, we maintain a stack S_e containing pairs of edges (e, f) or (f, e) that have been added to Π and to which we have not yet applied the implication rules. Further, we maintain a dynamic 2-dimensional spatial datastructure R_e for orthogonal range queries [MN90] that initially contains the endpoints of all edges of G. Each endpoint p is annotated with a list $L(p) \subseteq E$ of the edges incident to p.

We initialize the stacks according to the implication rules as follows. If e is an upward edge, then S_e is initialized with all edges in $\{(e,f) \mid f \in C_2(e)\} \cup \{(f,e) \mid f \in C_4(e)\}$. Similarly, if e is a downward edge, then S_e is initialized with all edges in $\{(e,f) \mid f \in C_1(e)\} \cup \{(f,e) \mid f \in C_3(e)\}$. The corresponding edges are also included in Π. To compute the edges for $C_i(e)$, we query R_e with $Q_i(e^-)$ and $Q_i(e^+)$, respectively. Finally, we remove the endpoints of all edges corresponding to the reported points from R_e.

For each edge e, we then proceed as follows. While the stack S_e is not empty, we pop the top element from the stack and apply the implication rules. For instance, assume that e is an upward edge and that we popped (f, e) from the stack, that is, e is strictly above f. Then we first compute all edges in $C_4(f)$ by querying R_e with $Q_4(f^-)$ and $Q_4(f^+)$ and enumerating the points in these regions. Whenever we find a point u we look up the corresponding edges $f_v = \{u, v\} \in L(u)$. Then we remove the endpoints of all edges f_v from R_e, insert the edges (f_v, e) into Π and put these edges onto the stack S_e. The cases, when e is a downward edge, or when we pop (e, f) from the stack, are handled in a similar fashion.

Note that there is an edge (e, f) in Π if and only if there is a sequence of edges $f'_1, \ldots f'_k$ such that

$$f'_1 \in C_2(e), f'_2 \in C_2(f'_1), \ldots, f'_k \in C_2(f'_{k-1}), f \in C_2(f'_k), \tag{6.1}$$

which is equivalent to

$$f'_k \in C_4(f), f'_{k-1} \in C_4(f'_k), \ldots, f'_1 \in C_4(f'_2), e \in C_4(f'_1)$$

due to the duality of the domination. Further, if there is are two edges (e, f) and (f, g), then by concatenating the sequences above, we have that (e, g) is also an edge in Π, that is, Π is the transitive closure of the implication rules.

Clearly, Π is a pre-embedding of G, that is, each edge (e, f) in Π encodes a necessary condition stating that e is strictly below f, since we only included edges according to the implication rules into Π. Hence, there is no orthogeodesic embedding for G, if Π contains a cycle corresponding to a conflicting set of necessary conditions. If Π contains a cycle, we can therefore conclude that no embedding exists and we are done.

Otherwise, assume that Π is acyclic. Then we show how to compute a combinatorial embedding from Π. If all pairs (e, f) of edges whose bounding boxes are intersected by a single vertical line are ordered with respect to Π, there is nothing to do. Otherwise, we iteratively construct a sequence of graphs $\Pi = \Pi_1, \ldots, \Pi_t$ such that Π_t corresponds to a combinatorial embedding of G. In iteration i we find a pair of edges (e_i, f_i) whose bounding boxes are intersected by a common vertical line and that are not ordered with respect to Π_i. Then we add an edge (e_i, f_i) to Π_i as well as all edges that are implied by (e_i, f_i), thus, obtaining Π_{i+1}. We stop if there is no pair of edges with the desired properties. To compute Π_{i+1} from Π_i we simply put the edge (e_{i+1}, f_{i+1}) on the stacks $S_{e_{i+1}}$ and $S_{f_{i+1}}$ and re-run the algorithm described above. Intuitively, this corresponds to adding another rule $e_{i+1} \prec f_{i+1}$ to the implication rules.

Assume that e is an upward edge. Since the edges (e_j, f_j) for $j \leq i$ do not correspond to necessary conditions, Equation (6.1) does not hold in its original version anymore for Π_i. Instead, there is an edge (e, f) in Π_i if and only if either Equation (6.1) holds as stated above or if a modified version holds with either $e = e_j$ or $f = e_j$ for some $1 \leq j \leq i$, that is, if there is an index j and a sequence of edges $f'_1, \ldots f'_k$ such that

$$f'_1 \in C_2(e_i), f'_2 \in C_2(f'_1), \ldots, f'_k \in C_2(f'_{k-1}), f \in C_2(f'_k)$$

or

$$f'_1 \in C_2(e), f'_2 \in C_2(f'_1), \ldots, f'_k \in C_2(f'_{k-1}), f_i \in C_2(f'_k) \,.$$

Intuitively, this means that every edge in Π_i is implied either by the original implication rules or by the newly added edges. As above, we have that Π_i is the transitive closure of the implication rules and the set of edges (e_j, f_j) for $1 \leq j \leq i$.

Suppose that $e := e_{i+1}$ and $f := f_{i+1}$ are unordered in Π_i and we create a cycle in Π_{i+1} when inserting the edge (e, f) and the edges implied by (e, f). Assume that e and f are both upward edges. The other cases are similar. All newly created edges are either of the form (e, g) such that there is a sequence of edges $f'_1, \ldots f'_k$ with

$$f'_1 \in C_2(f), f'_2 \in C_2(f'_1), \ldots, f'_k \in C_2(f'_{k-1}), g \in C_2(f'_k)$$

or of the form (g, f) such that there is a sequence of edges $f'_1, \ldots f'_k$ with

$$f'_1 \in C_4(e), f'_2 \in C_4(f'_1), \ldots, f'_k \in C_4(f'_{k-1}), g \in C_4(f'_k) \,.$$

Hence, we either find edges that must be embedded above e or edges that must be embedded below f, that is, the newly introduced edges are acyclic.

Since Π_{i+1} is the transitive closure of the implication rules and the edges (e_j, f_j) for $1 \leq j \leq i + 1$, a cycle in Π_{i+1} therefore implies the existence of an edge (f'_j, e) or and edge (f, f'_j) in Π_i for some $1 \leq j \leq k$. Assume that there is an edge $(f'_j, e) \in \Pi_i$. The case that there is an edge (f, f'_j) is analogous. First, observe that

$$f'_1 \in C_2(f), f'_2 \in C_2(f'_1), \ldots, f'_j \in C_2(f'_{j-1})$$

is equivalent to

$$f'_{j-1} \in C_4(f'_j), \ldots, f'_1 \in C_4(f'_2), \ldots, f \in C_4(f'_1),$$

by the duality of the domination. Since $f'_j \prec e$ the latter sequence of implications implies that $f \prec e$ in Π_i by the implication rules, that is Π_i must contain the edge (f, e), which contradicts the fact that we assumed e and f to be unordered. Therefore, the algorithm correctly computes a combinatorial embedding from the pre-embedding, given that the pre-embedding is acyclic.

It remains to show that the running time is bounded by $\mathcal{O}(n^2 \log n)$. For each edge, we initialize R_e in time $\mathcal{O}(n \log n)$. Subsequently, we perform at most n queries in amortized $\mathcal{O}(\log n)$ time each, since there are at most n edges incident to e in Π. We remove each point at most once from R_e without re-insertion. Therefore, the total time spent on R_e is bounded by $\mathcal{O}(n \log n)$ [MN90]. Since there are only $\mathcal{O}(n)$ edges and since each pair (e, f) of edges involves at most 4 stack operations, the running time of the algorithm is $\mathcal{O}(n^2 \log n)$. In order to extend the pre-embedding to obtain a regular orthogeodesic embedding, we must efficiently find pairs of edges whose bounding boxes can be pierced by a vertical line and that are not yet ordered in the current pre-embedding. In order to do this efficiently, we can maintain a list containing these edges. This list can be initialized in quadratic time. Then we iteratively pop edges from this list. We can efficiently test whether a newly popped pair of edges is comparable by additionally maintaining a marker for each pair of edges indicating whether they are comparable, for instance, in a quadratic-sized $m \times m$-matrix with 0/1-entries. Since each operation is added and removed from the list only once, these operations do not increase the asymptotic running time. □

Combining the results from the previous section we obtain the following.

Theorem 6.5. *Given a 4-planar graph G with n vertices, we can decide, in $\mathcal{O}(n^2 \log n)$ time, whether G admits an orthogeodesic embedding. Within the same time bound, we can construct such an embedding if one exists, or, if not, a proof of non-existence.*

Proof. First, we test whether G is admissible in linear time. If this is not the case, we return an in-admissible vertex to prove that the graph does not have a geodesic embedding. If the graph is admissible, we compute a pre-embedding of G and test whether it contains a cycle. The pre-embedding contains edges corresponding to necessary conditions for the existence of an orthogeodesic embedding. Every cycle thus corresponds to a set of conflicting requirements. If we additionally annotate each edge (e, f) with the edge (e', f') that implied this edge during the computation of the pre-embedding according to the implication rules, this proof is comprehensible and easy to check. Hence, if there is a cycle, we return the (annotated) pre-embedding to prove that no orthogeodesic embeddings exists. This can be done in $\mathcal{O}(n^2 \log n)$.

Otherwise, we extend the pre-embedding to a full combinatorial embedding as described in the proof of Lemma 6.3 in $\mathcal{O}(n^2 \log n)$ time. By Lemma 6.2 we obtain an orthogeodesic embedding from the combinatorial embedding in $\mathcal{O}(n^2)$ time. This concludes the proof. □

6.7 Concluding Remarks

We have introduced a new convention for drawing planar graphs, which we call the orthogeodesic drawing style. We require edges to be embedded as monotone orthogonal chains, that is, as shortest-possible orthogonal chains with respect to the Manhattan metric. This drawing style translates the good features of straight-line drawings into the rectilinear world:

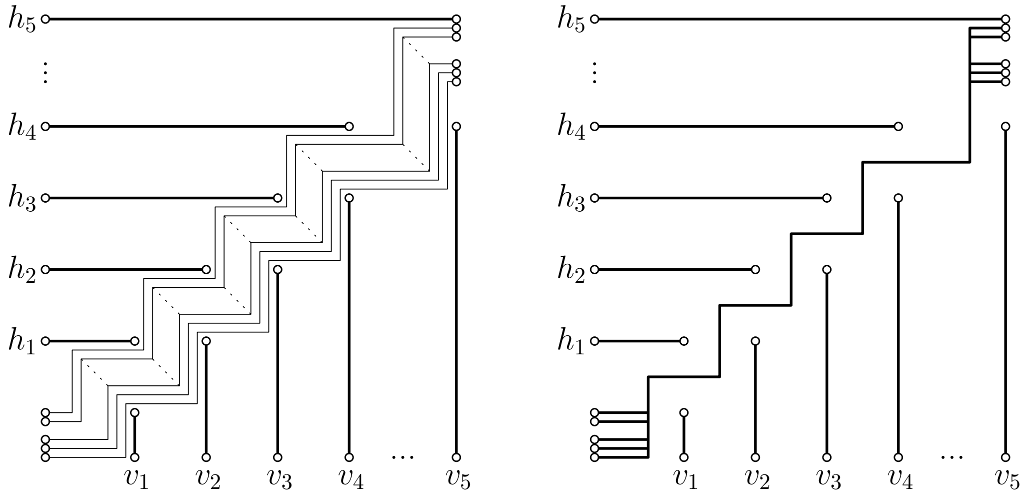


Figure 6.14: Instance of Labeled Orthogeodesic Matching with $3n$ edges and at least $2n^2$ bends and corresponding bundled orthogeodesic embedding with only $2n$ bends.

it combines monotonicity and shortness of edges with a limited number of slopes. We have investigated the complexity of several basic problems related to embedding graphs in this style. We have shown that we can efficiently decide whether a graph has a geodesic embedding on the grid, which is based on the observation that this is equivalent to 1-bend embeddability on the grid. If we are prescribed a set of points for placing the vertices, then the problem becomes NP-hard, even for subdivisions of cubic graphs. On the other hand, we have shown that point-set embeddability can be solved efficiently for cycles by proving an easy-to-check characterization of the point sets admitting an orthogeodesic polygonization and an efficient algorithm for computing such a polygonization if it exists.

The orthogeodesic embedding problem remains NP-hard if, in addition, the vertex–point correspondence is given—even for the case of perfect matchings. On the other hand, the problem can be solved efficiently for general graphs if we drop the grid limitation. We have provided an efficient certifying algorithm whose running time is almost worst-case optimal—up to a logarithmic factor.

Open problems An interesting set of open problem concerns the optimization of æsthetic criteria, such as the number of bends. Using a modification of the flow network by Tamassia, it is easy to see that we can compute a bend-minimal geodesic embedding on the grid if such an embedding exists. For sparse instances of the Labeled Orthogeodesic PSE problem, however, we do not know whether an efficient algorithm for minimizing the number of bends is within reach. We note that the sweepline algorithm we described for this case does not even provide an embedding with the minimum number of bends for the given combinatorial embedding. While the presented algorithm for the Labeled Orthogeodesic PSE is almost worst-case optimal, we could consider a modification of the drawing style in which sub-chains of the orthogeodesic chains representing the edges with the same embedding may be represented by a single *bundled* orthogeodesic chain as illustrated in Figure 6.14. In this drawing style, we can prove that the complexity of the embedding in terms of the number of the used straight-line segments is linear in the number of vertices of the graph. Thus, the lower bound we provided for the general case does not apply here. This raises the question if an orthogeodesic embedding in this drawing style can be computed in $o(n^2)$ time.

Further open problems concern orthogeodesic embeddability problems for various other classes of graphs. For instance, an interesting open problem concerns the question whether we can decide efficiently whether a given tree or a given outerplanar graph admits an orthogeodesic point-set embedding on a given point set. Finally, the orthogeodesic drawing style can also be considered for non-planar graphs.

Chapter 7

Orthogeodesic Embeddings of Trees

In the previous chapter we have studied different variants of the orthogeodesic embedding problem from a computational point of view. In this section, we adapt a combinatorial perspective and study which point sets admit orthogeodesic embeddings of all trees from a given class of trees in various orthogeodesic drawing styles. Let S be a set of N grid points in the plane, and let G be a graph with n vertices such that $n \leq N$. We study the following problem. Given a family of trees $\mathcal{F}$ what is the minimum value $f(n)$ such that every tree with n vertices in $\mathcal{F}$ admits an orthogeodesic point-set embedding on every grid-point set of size $f(n)$? We provide polynomial upper bounds on $f(n)$ for both planar and non-planar orthogeodesic point-set embeddings as well as for the case when edges are required to be L-shaped chains. This chapter is based on joint work with Emilio Di Giacomo, Fabrizio Frati, Radoslav Fulek and Luca Grilli [DGFF$^+$12].

7.1 Introduction

Let S be a set of N points in the plane, and let G be a graph with n vertices such that $n \leq N$. A *point-set embedding* of G on S is a drawing of G such that each vertex of G is drawn as a point of S. If, in addition, the drawing of G is crossing-free, that is, edges are not allowed to intersect in their interior, then the point-set embedding is called *planar*. Point-set embeddings are a classical subject of investigation in graph drawing from both an algorithmic and a combinatorial point of view. From the algorithmic point of view we are typically interested in deciding whether a given graph admits a point-set embedding on a given set of points. From the combinatorial perspective, on the other hand, we typically wish to characterize point sets that admit point-set embeddings for a whole class of graphs, such as trees or planar graphs. Different types of point-set embeddings have been defined depending on the desired type of drawing, that is, depending on how the edges are mapped to the plane. Further, point-set embeddings have been considered for various classes of graphs, such trees, planar graphs and outerplanar graphs as well as for various types of drawings, such as straight-line drawings and polyline drawings.

Previous work Several algorithmic results are known for point-set embeddings in which edges are required to be straight-line segments. Deciding whether a planar graph admits a straight-line planar point-set embedding on a given point set is an NP-complete problem [Cab06], while straight-line planar point-set embeddings of trees [BMS97] and outerplanar graphs [Bos02] can be computed efficiently. From the combinatorial perspective, Gritzmann et al. [GMPP91] prove that every planar graph with n vertices admits a planar straight-line point-set embedding on every set of n points in general position if and only if it is outerplanar. Kaufmann and Wiese show that every n-vertex planar graph admits a planar polyline point-set embedding on every set of n points with at most 2 bends per edge [KW02]. Colored versions of planar polyline point-set embeddings in which the points are colored and adjacent vertices must be mapped to points with different color have also been investigated [BDL08, DLT10]. Special research effort has also been devoted to the study *universal point sets* for planar graphs. A point set S is *universal* for a family $\mathcal{F}$ of graphs and for a type $\mathcal{D}$ of drawing if every graph in $\mathcal{F}$ admits a point-set embedding of type $\mathcal{D}$ on S. Every universal point set for straight-line planar drawings of planar graphs has size at least $1.235 \cdot n$ [Kur04], whereas there exist universal point sets of size $\frac{8}{9}n^2$ [Bra08]. For polyline point-set embeddings of planar graphs, on the other hand, there exist universal point sets of size n [ELLW10].

In this chapter we study *orthogeodesic point-set embeddings* on the grid. We already introduced orthogeodesic point-set embeddings and studied different variants of the orthogeodesic embedding problem from an algorithmic perspective. We proved that it is NP-hard to decide whether a planar graph with n vertices and maximum degree 4 admits an orthogeodesic point-set embedding on n points, while the problem can be solved efficiently for cycles. Further, we showed that, if the mapping between vertices and points is given and the bends are required to be at grid points, then the problem is NP-hard even for matchings, while the problem is polynomial-time solvable if bends need not be at grid points. Further, we will study a 2-colored version of the planar orthogeodesic point-set embedding in Chapter 8.

Contribution In contrast to Chapter 6, we consider orthogeodesic point-set embeddings *on the grid* from the combinatorial point of view in this chapter. Let P be a set of *grid points* in the plane, that is, $p = (i, j)$ with $i, j \in \mathbb{Z}$ for all $p \in P$. A set P of grid points with $x(p) \neq x(q)$ and $y(p) \neq y(q)$ for all $p, q \in P$ with $p \neq q$ is called a *general point set*. For different classes of trees $\mathcal{F}$ and different drawing styles $\mathcal{D}$ we study the value $f(n)$ such that *every* general point set is universal for orthogeodesic point-set embeddings of *all* trees in $\mathcal{F}$ using $\mathcal{D}$. The restriction to general point sets is necessary since there are arbitrarily large point sets that are not universal for orthogeodesic point-set embeddings of trees—for instance, a set of collinear points. Thus, without the restriction to general point sets $f(n)$ would not be well-defined for a large class of graphs. We consider both planar and non-planar orthogeodesic point-set embeddings as well as the case when edges can be arbitrary orthogeodesic chains and when edges are required to be L-shaped chains, respectively. Recall that an *L-shaped chain* is an orthogonal chain with only one bend, thus, it is an orthogeodesic chain with the minimum number of bends for general point sets. In the non-planar orthogeodesic drawing style we allow edges to be mapped to orthogeodesic chains that have a finite number of points in common, that is, two orthogeodesic chains are allowed to intersect, but they may not share common straight-line segments. Table 7.1 summarizes our results.

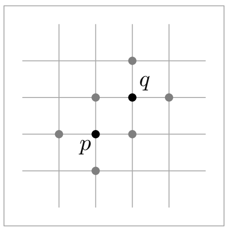
Organization This chapter is organized as follows. In Section 7.2, we study planar orthogeodesic point-set embeddings of trees without any further restriction. Then, we consider the case when edges are required to be L-shaped, that is, if they are allowed to have only one bend in Section 7.3. In Section 7.4, we drop the planarity constraint and we study L-shaped orthogeodesic point-set embeddings. Finally, we summarize our results and list open problems in Section 7.5.

Table 7.1: Summary of the results in the chapter. Each row corresponds to a family of trees $\mathcal{F}$ and each column corresponds to a type of drawing $\mathcal{D}$. The value in each entry is an upper bound to the minimum value $f(n)$ such that every n-vertex tree in $\mathcal{F}$ admits a point-set embedding of type $\mathcal{D}$ on every point set of size $f(n)$.

	L-Shaped		**Orthogeodesic**	
	Planar	Non-Planar	Planar	Planar 2-spaced
Caterpillars $\Delta = 3$	n [Th. 7.8]	n [Th. 7.8]	n [Th. 7.8]	n [Th. 7.1]
Trees $\Delta = 3$	$n^2{-}2n{+}2$ [Th. 7.6]	n [Th. 7.10]	n [Th. 7.3]	n [Th. 7.1]
Caterpillars $\Delta = 4$	$3n{-}2$ [Th. 7.7]	$n{+}1$ [Th. 7.11]	$\lfloor 1.5n \rfloor$ [Th. 7.4]	n [Th. 7.1]
Trees $\Delta = 4$	$n^2{-}2n{+}2$ [Th. 7.6]	$4n{-}3$ [Th. 7.9]	$4n$ [Th. 7.2]	n [Th. 7.1]

7.2 Planar Orthogeodesic Point-Set Embeddings

We start by considering planar orthogeodesic point-set embeddings of trees. Consider a general point set P such that both the horizontal and vertical distance between two distinct points is at least two, that is, $\min\{|x(p) - x(q)|, |y(p) - y(q)|\} \geq 2$ for all $p, q \in P$ with $p \neq q$. We call such a point set 2-*spaced*. In some sense, the point-set embedding problem is easier on 2-spaced point sets since these point sets do not contain pairs of points that are diagonally adjacent on the grid. Consider, for instance, two diagonally aligned points p and q such that $x(q) = x(p) + 1$ and $y(p) = y(q) + 1$ as illustrated in Figure 7.1a. Then there are only six grid points adjacent to p and q on the grid. However, if both vertices have degree four, then at least one grid point x must be used by two orthogeodesic chains incident to p and q as illustrated in Figure 7.1b. To see this, note that the horizontal and vertical grid lines through p and q, respectively may not contain any other vertex since we assumed that P is a general point set. On the other hand, all orthogeodesic chains incident to p and q, respectively, must use these lines. Assuming that p and q are connected by an L-shaped edge as illustrated in Figure 7.1b, the remaining six straight-line segments incident to p and q must pass five remaining points, which implies that one of the points, say x, is passed by two segments. This implies that two vertices with degree four may not be mapped to diagonally



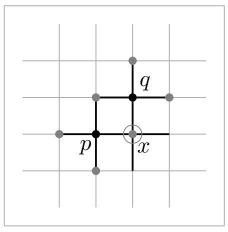


Figure 7.1: (a) Two points p and q that are diagonally adjacent on the grid and the six grid points adjacent to p and q. (b) Crossing resulting from mapping two vertices with degree four to p and q, respectively.

adjacent points on the grid. For instance, this implies that in any orthogeodesic embedding of a complete ternary tree, the internal vertices must be "interleaved" with the leaves in the sense that we may never place two internal vertices next to each other.

Since this restriction on the placement of the vertices makes the problem more complicated, we first consider the problem for 2-spaced point sets and we show that every tree with maximum degree 4 can be embedded on every general point set with n points using at most two bends per edge if we require that the horizontal and vertical distance of any two points is at least two. This implies that we can embed every tree with n vertices on every general point set P with n points whose points are not horizontally or vertically aligned, if neither vertices nor bends are required to be grid points.

Theorem 7.1. *Every tree with n vertices and with maximum degree 4 admits a planar orthogeodesic point-set embedding on every general point set P with n points such that* $\min\{|x(p) - x(q)|, |y(p) - y(q)|\} \geq 2$ *for all $p, q \in P$ with $p \neq q$.*

Proof. Let T be any tree with n vertices and maximum degree 4. We root T at any vertex r of degree at most 3. Inductively, we prove that T admits a planar orthogeodesic point-set embedding on every general point set P with n points such that the following invariants are maintained.

(T1) Each edge has two bends.

(T2) No edge intersects a half-line h arbitrarily chosen among the two horizontal and two vertical half-lines starting at r.

We will each subtree T' of T to a set of points contained in an axis-parallel rectangle. Since the half-line h does not intersect the drawing, we can use it to connect the root of T' to its parent using a straight-line segment on this half-line.

The claim is trivially true for $n = 1$. We inductively prove that T admits the required embedding for the case that no edge may intersect the horizontal half-line starting at r and directed rightward. The other constructions are analogous. Let $n_1 \geq 0$, $n_2 \geq 0$, and $n_3 \geq 0$ denote the number of vertices in the subtrees T_1, T_2, and T_3 rooted at children r_1, r_2, and r_3 of the root r of T, respectively. Let P_1 denote the set of the n_1 bottommost points of P. Let P_2 denote the set of the n_2 leftmost points of $P \setminus P_1$. Let p be the bottommost point of $P \setminus (P_1 \cup P_2)$. Let $P_3 = P \setminus (P_1 \cup P_2 \cup \{p\})$ as illustrated in Figure 7.2a. We embed r on p and we inductively embed T_i on P_i $(i = 1, 2, 3)$ such that no edge intersects the vertical half-line h_1 starting at r_1 in the upward direction, the horizontal half-line h_2 starting at r_2 in the rightward direction and the vertical half-line h_3 starting in r_3 in the downward direction. We connect r with r_1 by an orthogeodesic edge vertically attached to r and to r_1, respectively, and we connect the two vertical segments by an intermediate segment s on the horizontal line one unit above the top side of the bounding box of P_1. Further, we connect r with r_2 and r_3 analogously as illustrated in Figure 7.2b.

To see why the induction hypothesis holds, first note that the embeddings of $T_1, \ldots, T_3$ are crossing-free by induction-hypothesis and contained in disjoint axis-parallel rectangles. By choice of the point sets P_1, P_2 and P_3 no edge intersects the horizontal half-line h starting in r in the rightward direction by construction. Hence, it suffices to show that the resulting drawing is crossing free, that is, none of the edges connecting $r_1, \ldots, r_3$ to r are involved in any crossings. Clearly, these edge cannot cross each other by choice of $P_1, \ldots, P_3$ and the

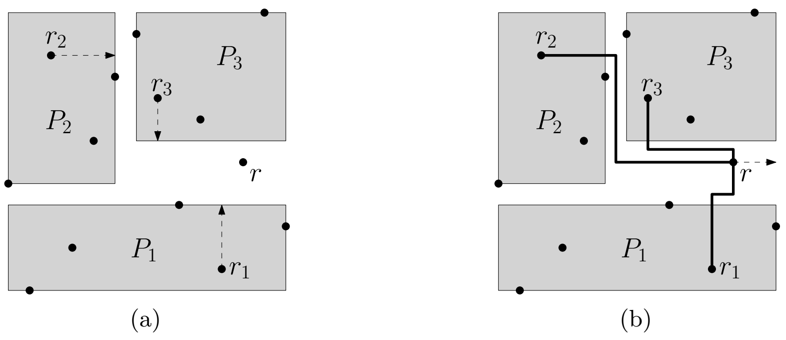


Figure 7.2: Planar orthogeodesic point-set embedding of a tree on a general point set with bends allowed to have half-integer coordinates.

construction of the corresponding orthogeodesic chains. Further, the straight-line segments incident to the vertices $r_1, \dots, r_3$ corresponding to the edges directed towards r, are mapped to the half-lines $h_1, \dots, h_3$ that are not crossed by any other edge by induction hypothesis. That is, there is no crossing in the bounding boxes of $P_1, \dots, P_3$, respectively. Next, consider the edge (r_1, r). The intermediate segment s is located on a horizontal grid line one unit above the highest point in P_1. Hence, this line does not contain any other point since we required $\min\{|x(p) - x(q)|, |y(p) - y(q)|\} \geq 2$ for all $p, q \in P$. Therefore, we can embed the edge as required by *(T1)* and *(T2)*. Similar arguments can be applied to the remaining edges, which concludes the induction step. $\square$

As an immediate consequence of Theorem 7.1 we obtain the following corollary for arbitrary point sets.

Corollary 7.1. *Let $P \subseteq \mathbb{R}^2$ be a set of points in the plane such $x(p) \neq x(q)$ and $y(p) \neq y(q)$ for all $p, q \in P$ such that $p \neq q$. Then every tree with maximum degree 4 has an orthogeodesic point-set embedding on P with at most two bends per edge.*

To see why Corollary 7.1 holds, we can consider a subdivision of the grid induced by the points in P. Let $x_1, \dots, x_n$ be the sorted sequence of the x-coordinates of the points in P and let $y_1, \dots, y_n$ be the sorted sequence of y-coordinates of the points in P. Let $\mathcal{G}$ be the grid induced by the horizontal and vertical lines through the points in P as well as by the horizontal lines $y = \frac{x_i + x_{i+1}}{2}$ and the vertical lines $x = \frac{y_i + y_{i+1}}{2}$ for $i = 1, \dots, n-1$. Then clearly, each point in $p \in P$ can be assigned a pair of integer coordinates (i_p, j_p) by numbering the horizontal grid-lines from bottom to top and the vertical grid lines from left to right such that $\min\{|i_p - i_q|, |j_p - j_q|\} \geq 2$. Then the corollary immediately follows from Theorem 7.1.

As another consequence of Theorem 7.1 we obtain the following theorem for general point sets on the grid without the restriction on the horizontal and vertical distance of the points.

Theorem 7.2. *Every tree with n vertices and with maximum degree 4 admits a planar orthogeodesic point-set embedding on every general point set with $4n$ points.*

Proof. We prove that any set P of $4n$ points contains a subset of n points such that no two points have a horizontal or vertical distance of less than two. The theorem then directly follows from Theorem 7.1. Let the points in P be $p_1, \dots, p_{4n}$ sorted from left to right. Let

P_1 consist of the $2n$ points p_{2i} $(1 \leq i \leq 2n)$ and the points in P_1 be $q_1, \ldots, q_{2n}$ sorted from bottom to top. Further, let P_2 consist of the n points q_{2i} $(1 \leq i \leq n)$. By construction the points in P_2 have the desired horizontal and vertical spacing and the theorem we can thus apply Theorem 7.1. □

For trees with maximum degree 3, however, we can improve this result by showing that every tree with this property admits a planar orthogeodesic point-set embedding on every general point set with n points using at most two bends per edge. Hence, every general point set with n points is universal for planar orthogeodesic point-set embeddings of trees with maximum degree 3.

Theorem 7.3. *Every tree with n vertices and with maximum degree 3 admits a planar orthogeodesic point-set embedding on every general point set with n points.*

Proof. Let T be a tree with maximum degree 3 and let P be a general point set with n points. We root T in a leaf r. Let w be the unique vertex incident to r. For a vertex v in T we denote the tree rooted in v by T_v. Then we construct a point-set embedding of T on P as follows. First, we embed r on the topmost point p_t of P and assign the subtree T_w rooted in w to the point set $P_w := P \setminus \{p_t\}$ and an axis-parallel rectangle R_w whose opposite corners are the left-bottom corner of the bounding-box of P and the point one unit below the right-top corner of the bounding-box of P. We connect r with the top border of R_w by drawing a vertical segment from p_t to the point p^* one unit below p_t as illustrated in Figure 7.3a with $r = p(v)$.

Next, we traverse T in a top-down fashion. When considering the subtree T_v of T rooted in v we suppose that T_v has already been assigned to a point set P_v and an axis-parallel rectangle R_v such that the following invariants hold.

(T1) The size of the tree T_v equals the size of the point set $|P_v|$.

(T2) The point set P_v is contained inside the rectangle R_v.

(T3) The parent $p(v)$ of vertex v lies outside of the rectangle R_v and is connected to R_v by a horizontal or vertical straight-line segment $\overline{p(v), p^*}$ such that p^* is located on the boundary of R_v.

(T4) Let T_u and T_v be two subtrees of T. If T_u is contained in T_v, then R_u is contained inside R_v. Similarly, if R_v is contained in R_u, then R_v is contained inside R_u. If neither T_u is contained in T_v nor T_v is contained in T_u, then $R_u \cap R_v = \emptyset$

(T5) Let T be a tree containing the edges $e = (p(v), v)$ and consider the straight-line segment $\sigma = \overline{p(v), p^*}$. Then σ is contained in a rectangle R_v such that T_v has been assigned to R_v if and only if T_v contains e.

Clearly, these invariants are satisfied after we have handled r as described above. Let v be an internal vertex of T. Since T has maximum degree 3, the subtree rooted in v has at most two children. Suppose that p^* is on the top side of R_v; the cases in which p^* is on the bottom, left, or right side of R_v can be discussed analogously. We consider two cases depending on the degree of v.

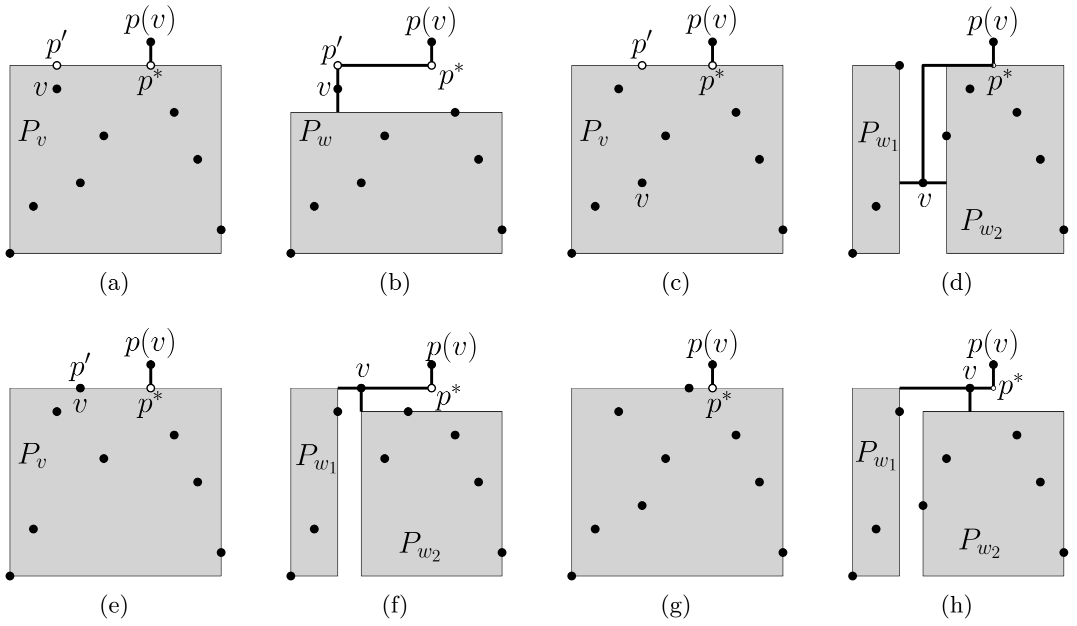


Figure 7.3: Embedding a tree with maximum degree 3 on a set of n points. (a) Embedding the root r. (b)–(c) Embedding s with exactly one child. (d)–(g) Embedding s with two children.

Case 1: First, suppose that v has one child w and consider Figures 7.3a and 7.3b. We embed v on the topmost point p_t of P_v and assign T_w to the point set $P_w := P_v \setminus \{p_t\}$ and to the rectangle R_w whose opposite corners are the left-bottom corner of R_v and the point one unit below the right-top corner of R_v. Let p^* be the point on the boundary of R_w that is vertically below $p(v)$ and let p' be the point on the boundary of R_w that is vertically above v. We connect $p(v)$ to v extending the horizontal straight-line segment $\overline{p(v)p^*}$ that we have already drawn by the invariant by the horizontal segment $\overline{p^*p'}$ and the horizontal segment $\overline{p'v}$. Finally, we draw a vertical segment connecting v to the top side of R_w as illustrated in Figure 7.3b.

Case 2: Next, suppose that v has two children w_1 and w_2. Let $P_{w_1} \subset P_v$ denote the point set composed of the leftmost $|T_{w_1}|$ points of P_v and let P_{w_2} denote the point set composed of the rightmost $|T_{w_2}|$ points in P_v. Further, we denote the single remaining point in $P_v \setminus (P_{w_1} \cup P_{w_2})$ by p. Let p^* be the point on the boundary of R_v vertically below $p(v)$ and let p' be the point on the boundary of R_v that is vertically above p. Then we assign T_{w_1} to the point set P_{w_1} and to the rectangle R_{w_1} whose opposite corners are the left-bottom corner of R_v and the intersection point between the top side of R_v and the vertical line one unit to the left of p. We consider two sub-cases depending on the line segment $\overline{p^*p'}$.

Case 2a: First, suppose that the straight-line segment $\overline{p^*p'}$ does not contain any point in P. This case is illustrated in Figures 7.3c and 7.3d. We embed v on p and we assign T_{w_2} to the point set P_{w_2} and the rectangle R_{w_2} whose opposite corners are the right-bottom corner of R_v and the intersection point between the top side of R_v and the vertical line one unit to the right of p. Then we connect $p(v)$ to v by extending the straight-line segment $\overline{p(s)p^*}$ by a horizontal segment $\overline{p^*p'}$ and a vertical segment $\overline{p'p}$. Finally, we draw a horizontal segment connecting v with the right side of R_{w_1} and we draw a horizontal segment connecting v with the left side of R_{w_2} as illustrated in Figure 7.3d.

Case 2b: Second, suppose that the segment $\overline{p^*p'}$ contains a point $q \in P$. This case is illustrated in Figures 7.3e–7.3h. We consider two sub-cases. First, suppose that $q = p$. Then we embed v on p and we assign T_{w_2} to the point set P_{w_2} and to the rectangle R_{w_2} whose opposite corners are the right-bottom corner of R_v and the point one unit below the intersection point between the top side of R_v and the vertical line through p. We connect $p(v)$ to v by extending the straight-line segment $\overline{p(s), p^*}$ by a horizontal segment $\overline{p^*p}$ as illustrated in Figure 7.3f. Second, suppose that $q \neq p$. Then we embed v on q and we assign assign T_{w_2} to the point set $P_{w_2} \setminus \{q\} \cup \{p\}$ and to the rectangle R_{w_2} whose opposite corners are the right-bottom corner of R_v and the intersection point between the horizontal line one unit below the top side of R_v and the vertical line through p. We connect $p(v)$ to v by extending the vertical segment $\overline{p(s)p^*}$ by the horizontal segment $\overline{p^*q}$.

Finally, in both cases, we draw a horizontal segment connecting v with the right side of R_{w_1} and we draw a vertical segment connecting v with the left side of R_{w_2} as illustrated in Figures 7.3f and 7.3h.

The case, when v is a leaf is handled similar to the case when v is an internal vertex with only one child.

Clearly, the invariants are maintained by the algorithm. The resulting drawing does not contain any crossings, since for each vertex v, the subtree T_v rooted in v is mapped to an axis-parallel rectangle that does not contain any vertex from $T - T_v$. Further, the constructed edges are orthogeodesic. Hence P admits and orthogeodesic point-set embedding of T □

Recall that a *caterpillar* is a tree such that by removing all leaves we are left with a path, called *spine*. In Theorem 7.2 we show that every tree with maximum degree 4 has a planar orthogeodesic point-set embedding on every general point set with $4n$ points. For caterpillars with maximum degree 4, however, this result is not tight.

Theorem 7.4. *Every caterpillar with n vertices and with maximum degree 4 admits a planar orthogeodesic point-set embedding on every general point set with $\lfloor 1.5n \rfloor$ points.*

Proof. Let C be a caterpillar with n vertices and with maximum degree 4 and let n_i denote the number of vertices of C with degree $i = 1, \ldots, 4$. Let P^* be a general point set with $\lfloor 1.5n \rfloor$ points. From P^* we arbitrarily choose a point set P of size $N = n + n_3 + n_4$ points on which we embed C. First, we show that $N \leq 1.5n$, which implies $N \leq \lfloor 1.5n \rfloor$ since N is a natural number. Suppose for contradiction that $n_3 + n_4 > n/2$. Since each vertex with degree at least 3 is incident to a leaf this yields $n_1 \geq n_3 + n_4$. Summing up we have $n \geq n_1 + n_3 + n_4 \geq 2(n_3 + n_4) > n$, a contradiction.

Next, we show how to embed C on P. Each vertex $v \in V$ is mapped to a point $\pi(v) \in P$. Let $S = (u_2, \ldots, u_{k-1})$ be the spine of C and let u_1 be a leaf incident to u_2 and let u_k be a leaf incident to u_{k-1}. By S^+ we denote the path $(u_1, \ldots, u_k)$. Then we consider the vertices u_i

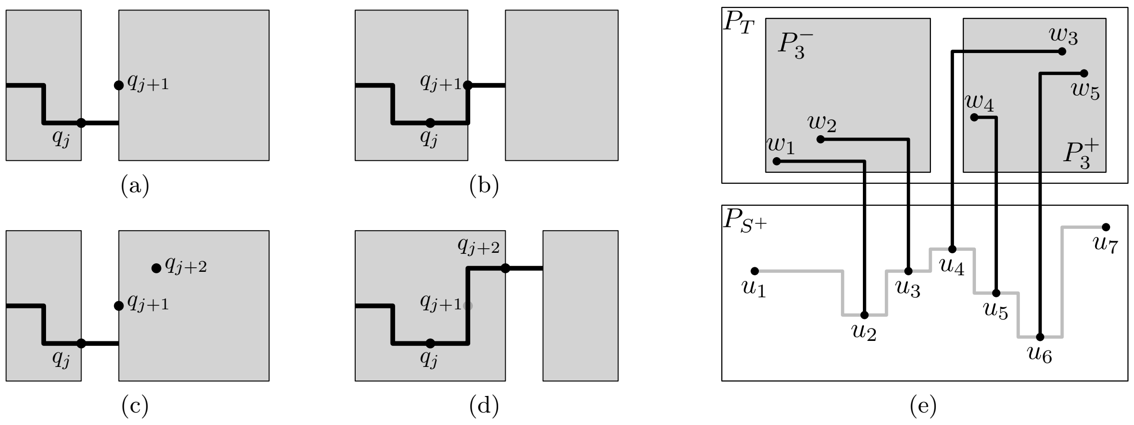


Figure 7.4: Embedding a caterpillar on a set of $\lfloor 1.5n \rfloor$ points. (a)–(d) Embedding the spine S^+. (e) Embedding the leaves in T.

for $i = 2, \dots, k-1$. If u_i has two adjacent leaves not in S^+, then we label one of them "top" and one of them "bottom". If u_i has only one adjacent leaf not in S^+, we arbitrarily label it "top" or "bottom". Let B and T be the sets of leaves of C that have been labeled "bottom" and by "top", respectively.

Let P_T be the subset of the highest $|T|$ points of P and let P_B be the subset of the lowest $|B|$ points. Further, let $Q = P \setminus (P_T \cup P_B)$ be the remaining points. By construction Q contains $t = n_2 + 2(n_3 + n_4) + 2$ points. We embed C on P as follows.

(S1) The leaves in T will be embedded on P_T, the leaves in B will be embedded on P_B and the vertices in S^+ will be embedded on a subset $P_{S^+} \subseteq Q$.

(S2) The spine will be embedded as an x-monotone chain such that u_i is left of u_{i+1} for all $1 \le i \le k-1$.

(S3) Edge $\{u_i, u_{i+1}\}$ occupies the horizontal segment incident to u_i on the right for all $1 \le i \le k-1$. If, additionally, the degree of u_i is at least 3, then edge $\{u_{i-1}, u_i\}$ occupies the horizontal segment incident to u_i on the left for all $2 \le i \le k-1$.

Let $q_1, \dots, q_t$ be the points in Q sorted from left to right. First, we map u_1 to the leftmost point q_1 in Q. Suppose, we have mapped $u_1, \dots, u_i$ for some $i < k$ and let $q_j = \pi(u_i)$. If u_{i+1} has degree 2, then we map u_{i+1} to q_{j+1} and we connect u_i and u_{i+1} by an L-shaped orthogeodesic chain composed of a horizontal segment incident to u_i and a vertical segment incident to u_{i+1} as illustrated in Figures 7.4a and 7.4b. If u_{i+1} has degree at least 3, then we map u_{i+1} to q_{j+2} skipping the point q_{j+1} in Q and we connect u_i by an orthogeodesic chain consisting of two horizontal segments incident to u_i and u_{i+1}, respectively, and a vertical segment in the column to the left of q_{j+2} as illustrated in Figures 7.4c and 7.4d. By construction, u_k is mapped to a point q_j such that $j \le n_2 + 2(n_3 + n_4) + 2$ since we only skipped points for vertices with degree at least 3.

Now we describe how to embed the leaves of T on P_T. The leaves in B are embedded on P_B analogously. Let $w_1, \dots, w_{|T|}$ be the vertices in T sorted such that their corresponding vertices on the spine are sorted from left to right and let T_i be the set of vertices in T that are incident to vertices u_j for $j < i$. For each i with $1 \leq i \leq k$ let P_i^- be the set of points in P_T to the left of $\pi(u_i)$ and let P_i^+ be the set of points in P_T to the right of $\pi(u_i)$, respectively, as illustrated in Figure 7.4e for vertex u_4. Each leaf w_i is mapped to a point $\pi(w_i)$ and is attached to the spine by an L-shaped orthogeodesic chain. We maintain the following invariant.

(L1) If w_i is incident to u_j and $|P_j^-| > |T_j|$, then w_i is mapped to the lowest point $p \in P_i^- \setminus \bigcup_{l=1}^{i-1}\{\pi(w_l)\}$ by an L-shaped orthogeodesic chain consisting of the vertical segment incident to $\pi(u_j)$ and the horizontal segment incident to p. Otherwise, w_i is mapped to the highest unused point in $P_i^+ \setminus \bigcup_{l=1}^{i-1}\{\pi(w_l)\}$ as illustrated in Figure 7.4e.

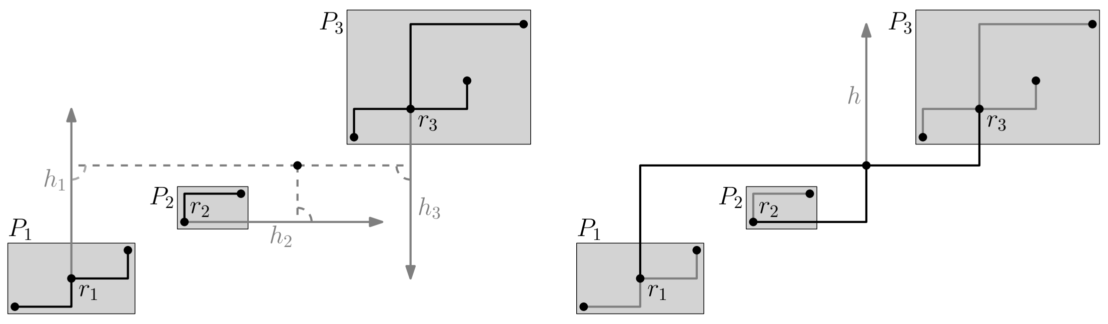
The resulting point-set embedding is orthogeodesic by construction. Planarity follows from the invariants as follows.

Due to invariants *(S1)* and *(S2)* the spine is mapped to an x-monotone chain such that the angle at vertices with degree at least 3 is 180 degrees. This implies that the spine does not cross itself and that the vertical segments incident to the vertices with degree at least 3 are unoccupied by the spine. Since, by invariant *(S1)*, we attached the leaves in T above the spine and the leaves in B below the spine, there cannot be a crossing between two edges incident to a leaf in T and a leaf in B, respectively. Suppose for contradiction that there is a crossing between two edges e_i and e_j incident to two leaves w_i and w_j in T, respectively. Without loss of generality we assume $i < j$. If $\pi(w_i) \in P_i^-$ and $\pi(w_j) \in P_j^+$ there cannot be a crossing by construction. If $\pi(w_i) \in P_i^- \subseteq P_j^-$ and $\pi(w_j) \in P_j^-$, then a crossing can only occur if $\pi(w_j) \in P_i^-$ and $\pi(w_j)$ is below $\pi(w_i)$, which contradicts invariant *(L1)*. Analogously, if $\pi(w_i) \in P_i^+$ and $\pi(w_j) \in P_j^+ \subseteq P_i^+$, then a crossing can only occur if $\pi(w_i) \in P_j^+$ and $\pi(w_i)$ is below $\pi(w_j)$, which contradicts invariant *(L1)*. Finally, if $\pi(w_j) \in P_i^-$ and $\pi(w_i) \in P_j^+ \subseteq P_i^+$, then this contradicts invariant *(L1)*, since w_i is only mapped to a point in P_i^+ if there is no unused point in P_i^-. Therefore, the embedding is crossing-free, which concludes the proof. □

7.3 Planar L-Shaped Orthogeodesic Pointset Embeddings

Next, we consider planar L-shaped orthogeodesic point-set embeddings of trees. First we prove that every tree with n vertices and with maximum degree 4 admits a planar L-shaped point-set embedding on every general point set with $n^2 - 2n + 2$ points. Every point set of this size contains a *diagonal* point set, which is universal for planar L-shaped point-set embeddings of trees with maximum degree 4. Let P be a point set and let $p_1, \ldots, p_n$ denote the points in P ordered by increasing x-coordinates. Then we refer to P as a *positive-diagonal point set* if $y(p_{i+1}) > y(p_i)$ for every $i = 1, \ldots, n-1$. Similarly, we refer to P as a *negative-diagonal point set* if $y(p_{i+1}) < y(p_i)$ for every $i = 1, \ldots, n-1$. If P is either a positive-diagonal point set or a negative-diagonal point set, then we call P a *diagonal point set.* First, we show that any diagonal point set is universal for L-shaped orthogeodesic point-set embeddings of trees with maximum degree 4.

Theorem 7.5. *Every tree with n vertices and with maximum degree 4 admits a planar L-shaped point-set embedding on every diagonal point set with n points.*



(a) Induction hypothesis (b) Induction step

Figure 7.5: Orthogeodesic point-set embedding of a tree with maximum degree 4 on a positive-diagonal point set.

Proof. Suppose that P is a positive-diagonal point set and let T be a tree with n vertices and with maximum degree 4. The case, when P is a negative-diagonal point set can be handled similarly. We root T at a vertex r with degree at most 3. By induction, we prove that T admits an orthogeodesic planar L-shaped point-set embedding on every diagonal point set with n points such that there is no edge overlapping or crossing a half-line h arbitrarily chosen among the two horizontal and two vertical half-lines starting at r.

In the base case $n = 1$ and the statement is trivially true. Suppose that the claim of the theorem is true for all $n' < n$. We show that T admits an orthogeodesic planar L-shaped point-set embedding on every diagonal point set P with n points such that no edge overlaps or crosses the vertical half-line h starting at r in the upward direction. The cases, when no edge overlaps the vertical half-line starting at r in the downward direction or the horizontal half-lines starting at r in the leftward or rightward direction, respectively, are handled analogously. Let $r_1, \dots r_k$ denote the children of r, that is, $k \leq 3$. Further, let n_i denote the number of vertices of the subtree T_i rooted in r_i for $i = 1, \dots, k$. If r has less than 3 children, we set $n_i = 0$ for $k < i \leq 3$. Let P_1, P_2, and P_3 be the point sets consisting of the bottommost n_1 points of P, the bottommost n_2 points of $P \setminus P_1$, and of the topmost n_3 points of P, respectively. Further, let p be the unique point in $P \setminus (P_1 \cup P_2 \cup P_3)$. By induction hypothesis, we can embed T_i on P_i for $1 \leq i \leq k$ as illustrated in Figure 7.5a such that no edge of T_1 intersects vertical half-line h_1 starting at r_1 in the upward direction, no edge of T_2 intersects the horizontal half-line h_2 starting at r_2 in the rightward direction and such that no edge of T_3 intersects the vertical half-line h_3 starting at r_3 in the downward direction. Since the bounding boxes of the point sets P_1, P_2 and P_3 are disjoint and since the geodesic chains corresponding to the edges of T_1, T_2 and T_3, respectively, are contained inside the bounding boxes of their respective point sets, the resulting embedding is crossing-free.

Then we embed r on p and we connect r to $r_1, \dots, r_k$ as illustrated in Figure 7.5b. That is, we connect r to r_i using the horizontal or vertical segment on h_i and the straight-line segment incident to r that is orthogonal to h_i for all $1 \leq i \leq k$. By the choice of h_i for $1 \leq i \leq k$, the edges are mapped to no-intersecting orthogeodesic chains and we do not use or cross the vertical half-line h starting at r in the upward direction. This concludes the induction step. $\square$

According to the Erdős-Szekeres theorem [ES35], every general point set with $n^2 - 2n + 2$ points contains either a positive-diagonal point set with n points or a negative-diagonal point set with n points. Hence, from Theorem 7.5 we immediately obtain the following theorem.

Theorem 7.6. *Every tree with n vertices and with maximum degree 4 admits a planar L-shaped point-set embedding on every general point set with $n^2 - 2n + 2$ points.*

For caterpillars with maximum degree 4 we can improve the bound of Theorem 7.6 as the following theorem shows.

Theorem 7.7. *Every caterpillar with n vertices and with maximum degree 4 admits a planar L-shaped point-set embedding on every general point set with $3n - 2$ points.*

Proof. Let C be a caterpillar with n vertices and with maximum degree 4 and let P be a general point set with $3n - 2$ points. Let $(u_2, \dots, u_{k-1})$ be the spine of C and let u_1 and u_k be two leaves of C adjacent to u_2 and to u_{k-1}, respectively. Let L denote the set of vertices of C containing all leaves of C, except u_1 and u_k. For $i = 1, \dots, k-1$ we let C_i denote the subtree of C induced by the vertices $u_1, \dots, u_i$ and by their adjacent leaves in $C - u_k$ and we let $C_k := C$. Observe that C_i is a caterpillar, for $i = 1, \dots, k$. By induction on i we prove that C_i admits a planar L-shaped point-set embedding on every general point set with $3|C_i| - 2$ points for all $i = 1, \dots, k$, such that the following invariant is satisfied.

(C1) The horizontal half-line starting at u_i directed rightward does not intersect any edge of the constructed drawing of C_i.

For $i = 1$ we have $|C_1| = 1$ and the induction hypothesis is trivially true. Suppose that the induction hypothesis is true for $i - 1$ and consider an arbitrary point set P_i with $3|C_i| - 2$ points. By P_{i-1} we denote the point set consisting of the leftmost $3|C_{i-1}| - 2$ points of P_i. Using the induction hypothesis, we can construct an orthogeodesic planar L-shaped point-set embedding of C_{i-1} on P_{i-1} such that the horizontal half-line starting at u_{i-1} in the rightward direction is not intersected by any edge of the constructed embedding. We distinguish three cases depending on the degree of u_i.

Case 1: First, assume that u_i is not adjacent to a leaf in L. Then, embed u_i on the rightmost point of P_i. Such a point exists since $|P_i \setminus P_{i-1}| = 3$. We connect u_i with u_{i-1} by an L-shaped edge using a horizontal straight-line segment incident to u_{i-1} that neither used nor crossed by any other edge by the induction hypothesis and a vertical straight-line segment attached to u_i.

Case 2: Second, assume that u_i is adjacent to exactly one leaf a_i and consider the three leftmost points of $P_i \setminus P_{i-1}$. These points exist since $|P_i \setminus P_{i-1}| = 6$. Then, either two of the three points are above the horizontal line h_{i-1} through u_{i-1} or two of the points are below h_{i-1}. Suppose that two of the points, say p_1 and p_2, are above h_{i-1}. The other case can be handled in a similar fashion. Without loss of generality we may assume that p_1 is to the left of p_2. Then, we embed u_i on the rightmost point p_2 and we embed a_i on the leftmost point p_1. Further, we connect u_i with u_{i-1} by an L-shaped edge horizontally attached to u_{i-1} and vertically attached to u_i and we connect u_i with a_i by an L-shaped edge horizontally attached to u_i and vertically attached to a_i.

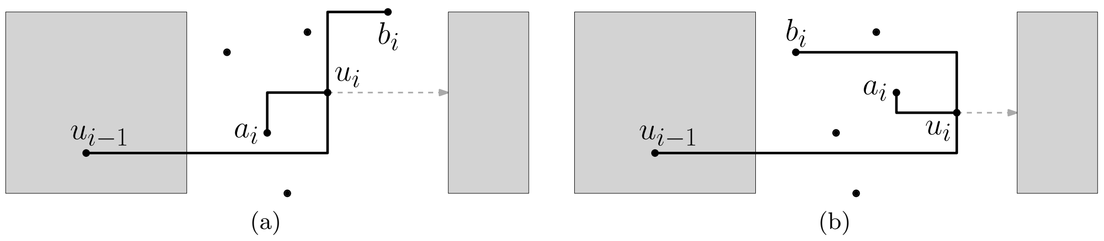


Figure 7.6: Planar L-shaped point-set embedding of caterpillars on general point sets. (a) $y(p_1) < y(p_2) < y(p_3)$. (b) $y(p_1) > y(p_2) > y(p_3)$.

Case 3: Third, assume that u_i is adjacent to two leaves a_i and b_i and consider the nine leftmost points of $P_i \setminus P_{i-1}$. These points exist since $|P_i \setminus P_{i-1}| = 9$. Then, either five of such nine points are above the horizontal line h_{i-1} through u_{i-1} or five are below. Suppose that five points $p_1, \dots, p_5$ are above h_{i-1}. The other case can be handled in a similar fashion. As a consequence of the Erdős-Szekeres-Theorem [ES35] every general point set with at least 5 points contains a diagonal point set with 3 points, hence the points $p_1, \dots p_5$ contain a diagonal point set with 3 points. Without loss of generality we may assume that $p_1, \dots, p_3$ form a diagonal point set and that $x(p_1) < x(p_2) < x(p_3)$. If $y(p_1) < y(p_2) < y(p_3)$ as illustrated in Figure 7.6a), that is, if the points $p_1, \dots, p_3$ form a positive-diagonal point set, then we embed u_i on p_2, a_i on p_1 and b_i on p_3. Similarly, if $y(p_1) > y(p_2) > y(p_3)$ as illustrated in Figure 7.6b), that is if the points $p_1, \dots, p_3$ for a negative-diagonal point set, then we embed u_i on p_3, a_i on p_2 and b_i on p_1. In both cases, we connect u_i with u_{i-1} by an L-shaped edge that is horizontally attached to u_{i-1} and vertically attached to u_i and we connect u_i to a_i by an L-shaped edge horizontally attached to u_i and vertically attached to a_i, and we connect u_i to b_i by an L-shaped edge vertically attached to u_i and horizontally attached to b_i as illustrated in Figures 7.6a and 7.6b, respectively.

Note that we did not use or cross the horizontal half-line starting at u_i in the rightward direction. Hence, the invariant *(C1)* is maintained, which concludes the induction. □

For caterpillars with maximum degree 3 we can improve this bound even further by showing that every such a caterpillar can be embedded on every general point set with n points using L-shaped edges.

Theorem 7.8. *Every caterpillar with n vertices and with maximum degree 3 admits a planar L-shaped point-set embedding on every general point set with n points.*

Proof. Let C be a caterpillar with n vertices and let P be a general point set consisting of n points $p_1, \dots, p_n$. Assume that the points are sorted such that $x(p_i) < x(p_{i+1})$ for all $1 \leq i \leq n-1$ and let $P_i := \{p_1, \dots, p_i\}$. Let $u_2, \dots, u_{k-1}$ denote the spine of C and let u_1 and u_k be two vertices adjacent to u_2 and u_{k-1}, respectively. Let C_i be the sub-tree of $C - u_k$ induced by the vertices $u_1, \dots, u_i$ and the leaves incident to these vertices for $1 \leq i \leq k-1$. Further, let $C_k := C$.

By induction on i we prove that we can find an planar L-shaped point-set embedding of C_i on P_j such that u_i such that the following invariants are maintained.

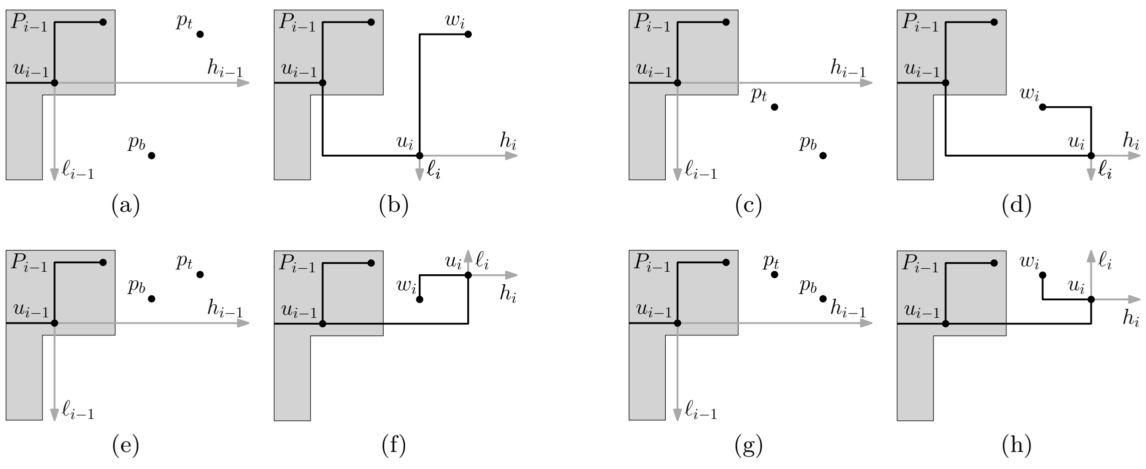


Figure 7.7: Orthogeodesic planar L-shaped point-set embedding of a caterpillar with maximum degree 3.

(C1) The size of $|P_j|$ equals the size of C_i, that is, $j = |C_i|$.

(C2) The vertex u_i is mapped either to p_j or to p_{j-1}.

(C3) Both the horizontal half-line h_i starting at p_j in the rightward direction as well at least one vertical half-line ℓ_i starting at p_j either in the upward or in the downward direction do not intersect the drawing of C_i.

The induction hypothesis is trivially true for $i = 1$. We map u_1 to p_1 and let h_1 denote the horizontal half-line starting at p_1 in the rightward direction. Further, we let ℓ_1 denote the vertical half-line starting at p_1 in the downward direction. Now, suppose that the induction hypothesis is true for $i - 1$ and consider the caterpillar C_i. By the induction hypothesis, we can find an planar L-shaped point-set embedding of C_{i-1} on P_j such that $j = |C_{i-1}|$. Assume, without loss of generality, that ℓ_{i-1} denotes the vertical half-line starting at u_{i-1} in the downward direction and that ℓ_{i-1} does not intersect the drawing of C_{i-1} as illustrated in Figure 7.7a and suppose that u_{i-1} is mapped to p_c such that $c \in \{|C_i|, |C_{i-1}|\}$. Again we distinguish two sub-cases depending on the degree of u_i.

Case 1: First, suppose that u_i has degree at most two, that is it is adjacent to at most two vertices u_{i-1} and, possibly, u_{i+1} if it exists. Then we map u_i to p_{c+1} and we connect u_i to u_{i-1} by an L-shaped edge that is horizontally attached to u_{i-1} and vertically attached to u_i. Clearly the invariants are maintained.

Case 2: Second, suppose that u_i has degree three, that is $i < k$ and u_i is adjacent to two vertices u_{i-1} and u_{i+1} as well as an additional leaf w_i. Note that $|C_i| = c + 2$, that is, $|C_i| - |C_{i-1}| = 2$. Let p_b and p_t denote the two vertices in $P_{|C_i|} \setminus P_{|C_{i-1}|}$ such that $y(p_b) < y(p_t)$, that is p_b is below p_t. We distinguish two sub-cases.

Case 2a: First, suppose that p_b is below the vertical half-line h_{i-1} as illustrated in Figures 7.7a and 7.7c. Then we map u_i to p_b and we map w_i to p_t. Further, we connect u_{i-1} to u_i by an L-shaped edge that is vertically attached to u_{i-1} and horizontally attached to u_i and we attach w_i to u_i by an L-shaped edge that is vertically attached to u_i and horizontally attached to w_i as illustrated in Figures 7.7b and 7.7d. This way, the horizontal half-line h_i starting at u_i in the rightward direction as well as the vertical half-line ℓ_i starting at u_i in the downward direction do not intersect the constructed embedding. Hence, the invariants are maintained.

Case 2b: Second, suppose that p_b is above the vertical half-line h_{i-1}, that is, p_t is also above h_{i-1} as illustrated in Figures 7.7e and 7.7g. We map u_i to the rightmost point of p_b and p_t and we map w_i to the leftmost point of p_b and p_t, respectively. Further, we connect u_i to u_{i-1} by an L-shaped edge that is horizontally attached to u_{i-1} and vertically attached to u_i and we connect w_i to u_i by an L-shaped edge that is horizontally attached to u_i and vertically attached to w_i as illustrated in Figures 7.7f and 7.7h.

Note that the newly constructed edges do not intersect. Clearly, the constructed embedding does not intersect the horizontal half-line h_i starting at u_i in the rightward direction and the vertical half-line ℓ_i starting at u_i in the upward direction. Hence, the invariants are maintained. This concludes the induction. □

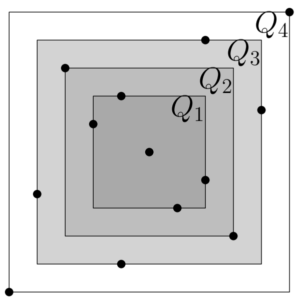
7.4 Non-Planar L-Shaped Orthogeodesic Point-Set Embeddings

Next, we consider non-planar L-shaped orthogeodesic point-set embeddings on general point sets P. In a non-planar L-shaped orthogeodesic embedding we require that edges are mapped to orthogeodesic chains such that each chain contains exactly two points from P, that is, its endpoints, and has only a finite number of points in common with any other chain. That is, orthogeodesic chains may cross but are not allowed to overlap. We start by showing that every tree with n vertices as a non-planar L-shaped orthogeodesic point-set embedding on every general point set with $4n-3$ points.

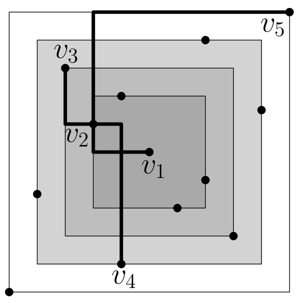
Theorem 7.9. *Every tree with n vertices and with maximum degree 4 admits a non-planar L-shaped point-set embedding on every general point set with $4n-3$ points.*

Proof. Let $T=(V,E)$ be a tree with n vertices and let P be a general point set with $4n-3$ points. Let T be rooted in a leaf $r \in V$ and let the vertices of T be labeled $r=v_1,\ldots,v_n$ according to a depth-first search in T. Let $Q_n=P$. For $n \geq i \geq 1$, let P_i consist of the points on the boundary of the bounding box of Q_i, and for $n \geq i \geq 2$ let $Q_{i-1}=Q_i \setminus P_i$ as illustrated in Figure 7.8a. Since the boundary of the bounding box of a general point set contains at least two and at most four points, and since P contains $4n-3$ points, we have that each set P_i contains at least two and at most four vertices, except for P_1, which contains at least one vertex.

We embed T using L-shaped orthogeodesic chains such that vertex v_i is mapped to a point in P_i for all $1 \leq i \leq n$. We start by mapping the root v_1 to an arbitrary point $p^* \in P_1$. Suppose we have embedded all vertices $v_1,\ldots v_i$ for some $i \geq 1$ and we would like to embed v_{i+1}. Since the vertices are ordered according to a depth-first search, we have already embedded the parent v_j of v_{i+1}. Further, the vertices $v_1,\ldots,v_j$ have been embedded inside the bounding box



(a) Point set



(b) Embedding v_2v_5

Figure 7.8: Non-planar L-shaped point-set embedding of a tree with maximum degree 4.

of the point set Q_i which in the interior of the bounding box of the points in Q_{i+1}. Since v_j has degree at most 4 and since, we have not yet mapped v_{i+1}, at least one of the segments incident to v_j in the drawing is unoccupied by the drawing. Without loss of generality we may assume that the vertical segment above v_j is unoccupied (otherwise we can rotate the instance accordingly). By construction the points in P_{i+1} are on the bounding box of Q_{i+1}, which contains Q_i in its interior. Hence, P_{i+1} contains a point p_t on the top side of the bounding box of Q_{i+1}. Then we map v_{i+1} to p_t and we connect it to v_j by an L-shaped edge that is vertically attached to v_j and horizontally attached to p_t as illustrated in Figure 7.8b. □

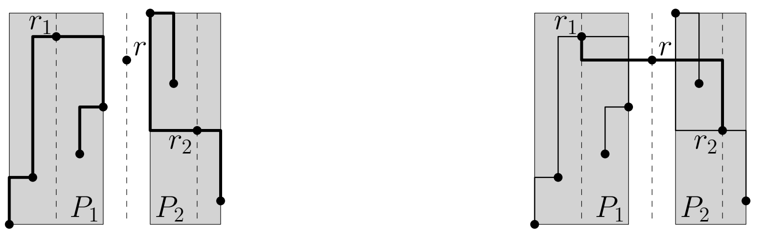
Next, we improve on this by showing that a general point set of size n allows an L-shaped point-set embedding for the class of trees with n vertices and maximum degree 3.

Theorem 7.10. *Every tree with n vertices and with maximum degree 3 admits a non-planar L-shaped point-set embedding on every general point set with n points.*

Proof. Let T be a tree with n vertices and with maximum degree 3 and let P be a general point set with n points. Assume that T is rooted at a vertex r with degree at most 2. By induction on n we prove that we can find an L-shaped point-set embedding of T on P such that none of the edges occupies the vertical line through r.

If $n = 1$, we map the single vertex of T to the single point in P and we are done. Suppose that the induction hypothesis holds for all $n' < n$. Let $n_1 \geq 0$ and $n_2 \geq 0$ denote the number of vertices in the subtrees T_1 and T_2 rooted at the children r_1 and r_2 of r, respectively. Further, let P_1 and P_2 be the point sets consisting of the leftmost n_1 points and the rightmost n_2 points of P, respectively, as illustrated in Figure 7.9a. Let p be the unique point of P not in P_1 and not in P_2. Then we embed r on p. By induction we can find an L-shaped point set embedding of T_1 on P_1 and of T_2 on P_2 such that the vertical line through r_1 and r_2 is unoccupied by any edge of the resulting drawings, respectively. Then, we connect r to r_1 by an L-shaped edge that is horizontally attached to r and vertically attached to r_1. Similarly, we connect r to r_2 by an L-shaped edge that is horizontally attached to r and vertically attached to r_2 as illustrated in Figure 7.9b. Since the constructed edges are attached to r horizontally and since the embeddings of T_1 and T_2 are contained in the bounding boxes of their respective point sets, which do not intersect the vertical line through r, the maintain the invariant as claimed, which concludes the induction step.

□



(a) induction hypothesis (b) induction step

Figure 7.9: Non-planar L-shaped point-set embedding of a tree on a general point set.

For caterpillars with maximum degree 4 we can further improve this by showing that every general point set with $n+1$ points admits an orthogeodesic L-shaped point-set embedding of every caterpillar with n vertices and with maximum degree 4.

Theorem 7.11. *Every caterpillar with n vertices and with maximum degree 4 admits a non-planar L-shaped orthogeodesic point-set embedding on every general point set with $n+1$ points.*

Proof. Let P be a general point set with $n+1$ points. Let C be a caterpillar with maximum degree 4 and let $(u_2, \dots, u_{k-1})$ denote the vertices of its spine. Further, let S^+ denote the path $(u_1, u_1, \dots, u_k, u_k)$ where u_1 and u_k are two leaves incident to u_2 and u_{k-1}, respectively. We embed C on P using L-shaped orthogeodesic chains for the edges such that the following invariants are maintained.

(S1) The spine is embedded as a monotone chain starting in the leftmost point in P.

(S2) The spine leaves each vertex along the horizontal segment to its right and enters each vertex along a vertical segment either above or below it.

(S3) All but possibly one point to the left of u_i are occupied by the vertices u_j for $i < j$ and the leaves adjacent to these vertices.

By applying (S3) to u_{k+1} it is clear that $n+1$ points are sufficient for the embedding.

First, we embed u_1 on the leftmost point in P. Suppose we have mapped all vertices $u_1, \dots, u_i$ for some $0 \le i \le k$. Let u_i be mapped to p_j and let P_i^+ be the remaining points to the right of u_i that are not yet occupied by a point.

In order to embed u_{i+1} as well as the leaves incident to it, we distinguish four cases.

Case 1: u_{i+1} has degree at most two. Let p be the leftmost point in P_i^+. We map u_{i+1} to p and connect it to u_i an L-shaped orthogeodesic chain as illustrated in Figures 7.10a and 7.10b.

Case 2: u_{i+1} has degree 3. Let w be a leaf incident to u_{i+1}. Let p_1 be the leftmost point in P_i^+ and let p_2 be the leftmost point in P_i^+ to the right of p_1. We map u_{i+1} to p_2 and connect it to u_i by an L-shaped orthogeodesic chain starting with a horizontal segment in u_{i+1}. Further, we map w to p_1 and connect it to u_{i+1} by the horizontal segment incident to p_2 to the left and the vertical segment incident to p_1 as illustrated in Figures 7.10c and 7.10d.

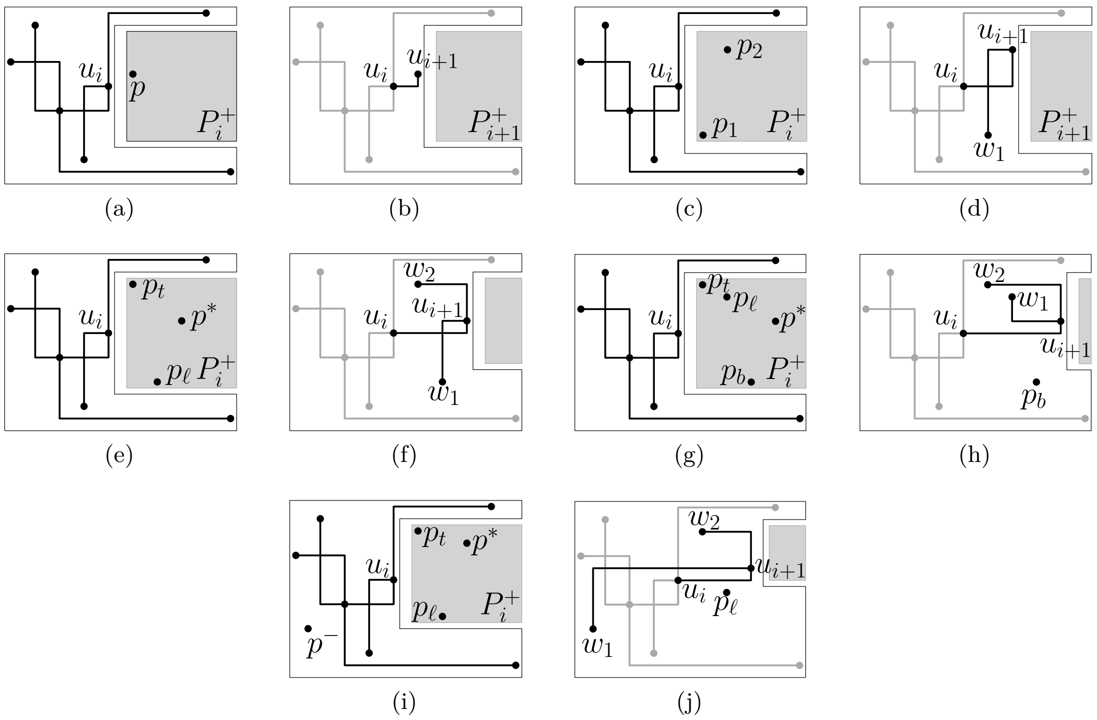


Figure 7.10: Embedding a caterpillar on $n + 1$ points using L-shaped edges.

Case 3: u_{i+1} has degree 4 and there is no unoccupied point to the left of u_i. Let w_1 and w_2 be two leaves incident to u_{i+1}. Recall that we assumed that the vertex u_i be mapped to a point p_j. Let p^* be the leftmost point in P_i^+ with the following property.

> Either *(i)* p^* is above p_j and p^* contains two distinct points p_ℓ and p_t to its left and above, respectively, as illustrated in Figures 7.10e and 7.10g or, similarly, *(ii)* p^* is below p_j and p^* contains two distinct points p_ℓ and p_b to its left and below, respectively.

Let Q denote the set of the leftmost 4 points in P_i^+. We claim that Q contains a point p^* with the desired property. Let p_t be the topmost point in Q and let p_b be the bottommost point in Q. Further, let p_ℓ be the leftmost point such that $p_\ell \neq p_t, p_b$ and let q be the remaining point. By construction, q has the desired properties. Hence, there can be at most three points to the left of p^*.

We assume that p^* is above p_j as illustrated in Figures 7.10e and 7.10g, respectively. The case when p^* is below p_j is analogous. First, we consider the case that there are only two points in P_i^+ to the left of p^*, namely a point p_t above p^* and a point p_ℓ left of p^* as illustrated in Figure 7.10e. We map u_{i+1} to p^* and connect it to u_i by an L-shaped orthogeodesic chain consisting of the horizontal segment incident to u_i and the vertical segment incident to u_{i+1}. Further, we map w_1 to p_ℓ and connect it to p^* by an L-shaped orthogeodesic chain consisting of the horizontal segment incident to p^* and the vertical segment incident to p_ℓ. Further, we map w_2 to p_t and connect it by the respective orthogeodesic chain as illustrated in Figure 7.10f.

Next, we consider the case that there are three points to the left of p^* as illustrated in Figure 7.10g. Let Q, p_t, p_b and p_ℓ be chosen as described above. We embed u_{i+1} on p^*, w_1 on p_ℓ and w_2 on p_t as in the above description and we leave the point p_b to the left of p^* unused as illustrated in Figure 7.10h.

Case 4 u_{i+1} has degree 4 and there is a single unoccupied point p^- to the left of u_i. This case is analogous to the Case 3, except that we do not require that p^* contains a point p_ℓ to its left in P_i^+, since p^- will substitute p_ℓ. Note that, as in Case 3, one single point to the left of p^* may remain unoccupied as illustrated in Figures 7.10i and 7.10j.

Clearly, by construction of the embedding the invariants are maintained, which implies that at most one of the points will not be used by the embedding. □

7.5 Concluding Remarks

In this chapter we studied orthogeodesic point-set embeddings of trees on the grid. For various types of drawings $\mathcal{D}$ and various families of trees $\mathcal{F}$ we proved upper bounds on the minimum value $f(n)$ such that every n-vertex tree in $\mathcal{F}$ admits a point-set embedding of type $\mathcal{D}$ on every general point set of size $f(n)$ on the grid. The restriction to general point sets was motivated by the fact that $f(n)$ would be infinity for most of the interesting classes of graphs if we allowed arbitrary point sets. We showed that every tree with n vertices and maximum degree 4 admits a planar orthogeodesic point-set embedding on every general point set with $4n$ points on the grid. This implies that every tree with n vertices admits a planar orthogeodesic point-set embedding on every general point set if we do not require the orthogeodesic chains to be on the grid. We then improved on this by considering trees with maximum degree 3 and caterpillars with maximum degree 4. We showed that every caterpillar with maximum degree 4 admits an orthogeodesic point-set embedding on every general point set with $1.5n$ points, whereas every tree with maximum degree 3 admits an orthogeodesic point-set embedding on every general point set with n points.

Next, we considered planar L-shaped orthogeodesic point-set embeddings. While all of the constructions up to this point used at most two bends on each edge, L-shaped orthogeodesic chains are optimal with respect to the number of bends for an orthogeodesic point-set embedding of a graph on a general point set. We showed that every tree with n vertices and maximum degree 4 admits a planar orthogeodesic point-set embedding on every point-set with $n^2 + 2n + 2$ points. We improved on this for caterpillars with maximum degree 3 and 4 by showing that every caterpillar with maximum degree 4 admits a planar orthogeodesic point-set embedding on every general point set with $3n - 2$ points whereas a caterpillar with maximum degree 3 admits a planar orthogeodesic point-set embedding on every general point set with n points.

Finally, we considered non-planar orthogeodesic point-set embeddings which allow edges to cross and we proved that every tree with n vertices and maximum degree 4 admits an orthogeodesic embedding on every general point set with $4n - 3$ points, which we improved to $n + 1$ and n for caterpillars with maximum degree 4 and trees with maximum degree 3, respectively.

Interestingly, we could not prove a sub-quadratic bound on $f(n)$ for planar L-shaped point-set embeddings or a non-trivial lower bound on $f(n)$. On the other hand, an extensive

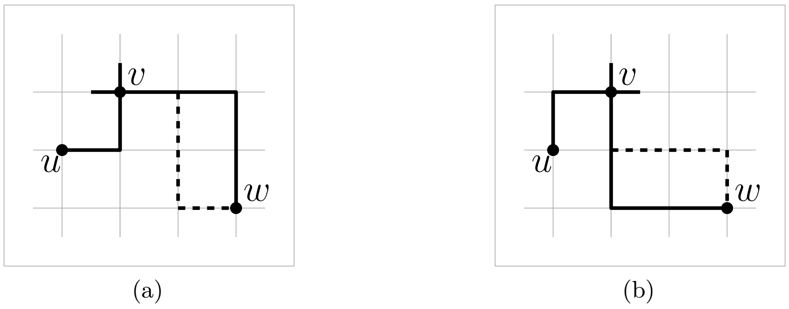


Figure 7.11: A path of three vertices u, v and w such that v has degree 4. While the embedding of the edge uv does not affect whether the edge vw is connected to w by a horizontal or a vertical segment if two bends are allowed, as illustrated by the solid and dashed edge, respectively, there is only one embedding of the edge vw depending on the embedding of the edge uv if only one bend is allowed.

experimental analysis did not reveal any tree with n vertices that did *not* be admit a planar L-shaped orthogeodesic point-set embedding on any of the tested point sets with n points. For the experimental analysis, we implemented an algorithm for orthogeodesic point-set embeddings based on transforming the problem into a SAT problem and using the MiniSat solver [ES05] to compute a layout or prove that no layout exists. But what makes these embeddings so hard to analyze? On the one hand, we already mentioned in Section 7.2 that we are not allowed to map two vertices with degree 4 to diagonally adjacent points on the grid. In fact, we could prove much better bounds for the non-planar L-shaped point-set embeddings, so at least part of the difficulty results from this observation. On the other hand, allowing two bends on each edge seemed to also alleviate the situation, while not allowing two bends introduces an addition difficulty due to the following observation.

Consider a path consisting of three vertices u, v and w such that v has degree 4. The vertices are mapped to the grid as illustrated in Figure 7.11a. There are two possibilities of embedding an L-shaped edge between u and v as illustrated in Figure 7.11a and 7.11b, respectively. Whereas the choice of this embedding does not affect whether the edge vw is attached to w by a horizontal or a vertical straight-line segment if two bends are allowed, there is only one possible embedding left if only one bend is allowed, and this embedding depends on the choice of the embedding of uv. Thus, the restriction to L-shaped edges introduces an additional level of dependency between the embedding of individual edges, which makes the problem considerably more complicated.

Open Problems Since n is a trivial lower bound for $f(n)$ in all considered variants of the problem and since the upper bounds we provided are larger than n for some of the considered variants, it is an interesting topic for future research to close the gap between n and $f(n)$. The gap is especially large for planar L-shaped point-set embeddings for which we only proved a quadratic upper bound. Hence it would be interesting to come up with a sub-quadratic upper bound or a non-trivial lower bound.

Further, we restricted our attention to trees, but we may consider the same problem for different classes of graphs. In a preliminary study we could prove that there are general point sets with n points that are not universal for planar orthogeodesic point-set embeddings of outerplanar graphs with n vertices, regardless of the number of bends we allow per edge and regardless of whether we require the embedding to be outerplane or not. Hence, this proves a non-trivial lower bound on $f(n)$ for outerplanar graphs. Given that not all general point sets are universal for all outerplanar graphs, however, it is an interesting question which point sets—if any—are universal for outerplanar graphs. Further, it is an interesting question to determine the minimum size $f(n)$ such that all outerplanar graphs admit an orthogeodesic point-set embedding on all general point sets with $f(n)$ points.

Chapter 8

Hamiltonian Orthogeodesic Alternating Paths

In Chapter 6 we studied different variants of orthogeodesic point-set embedding problem. We considered the two extreme situations in which the vertices of the network were allowed to be mapped to either any point of the given point set or merely to a uniquely specified point. In this chapter we consider an intermediate scenario for path networks, where a single vertex of the path can be embedded on several but not all points of the given point set. This kind of constraint is typically modeled by considering points as being colored and requiring adjacent point to embedded on points with different colors.

Given a set of red and blue points, an orthogeodesic alternating path is a path such that each edge is an orthogeodesic chain connecting points of different color and such that no two edges cross. We consider the problem of deciding whether there exists a *Hamiltonian* orthogeodesic alternating path, that is, an orthogeodesic alternating path visiting all points. We prove that every general point set containing the right number of blue and red points contains a Hamiltonian orthogeodesic alternating path and we present an efficient an $O(n \log^2 n)$-time algorithm for finding such a path. Further, we show that the problem is NP-complete if the path must be on the grid. On the other hand, we show that we can approximate the maximum number of vertices of an orthogeodesic alternating path on the grid by a factor of 3, whereas we present a family of point sets with n points that do not have a Hamiltonian orthogeodesic alternating path with more than $n/2+2$ points. Additionally, we show that it is NP-complete to decide whether a given set of red and blue points on the grid admits an orthogeodesic perfect matching if horizontally aligned points are allowed. This contrasts a recent result by Kano [Kan09] who showed that this is possible on every general point set. This chapter is based on joint work with Emilio Di Giacomo, Luca Grilli, Giuseppe Liotta and Ignaz Rutter [DGGK$^+$12].

8.1 Introduction

The study of point-set embedding problems can be motivated, for instance, from applications in circuit layout, where the task is to lay out a given graph on a circuit board subject to certain constraints on the routing of the edges and the placement of the vertices. Motivated by a limited resolution, special emphasis is typically put on embeddings on the grid. We have already considered two variants of the orthogeodesic embedding problem in which the vertices of the network could be placed on either all given points or a uniquely specified point. However, typically, we are not given the exact placement of the vertices and vertices may not be placed arbitrarily. In this chapter, we consider a special restriction on the placement of the vertices, which is in fact probably one of the simplest and least restrictive scenarios by assuming that the points have different characteristics and adjacent vertices may not be mapped to points with the same characteristic. In order to model this scenario, we assume that the points are colored using two colors. In previous work on this kind of problem the points are usually colored red and blue.

Let R be a set of red points and let B be a set of blue points such that $|R| \leq |B|$[1]. The point set $P = R \cup B$ is called *equitable* if $|B| - |R| \leq 1$ and it is called *balanced* if $|B| = |R|$. The color of a point $p \in P$ is denoted by $c(p)$. An *alternating path* on a set of red and blue points P is a sequence of points $p_1, \ldots, p_h$ that is alternatingly colored red and blue, such that p_i is connected to p_{i+1} $(i = 1, \ldots, h-1)$ by a rectifiable curve. Throughout this chapter we only consider the case when the curves corresponding to the edges of the path are not allowed to intersect. If the curves along the path are straight-line segments, then the path is called a *straight-line alternating path*; if the curves are orthogeodesic chains, then the path is called an *orthogeodesic alternating path*. Given a set of points P, an alternating path is called *Hamiltonian* if it contains all points in P. Clearly a Hamiltonian alternating path can only exist on an equitable point set. In this chapter, we study combinatorial and algorithmic aspects of Hamiltonian orthogeodesic alternating paths on and off the grid. We note that a given equitable point set P with n points admits a Hamiltonian orthogeodesic alternating path if and only if a path with n vertices admits an embedding on P such that adjacent vertices are mapped to points with different colors.

Previous work The problem of computing an alternating path on a given equitable set of points in general position is a classical subject of investigation in the computational geometry field and several papers are devoted to Hamiltonian alternating paths. Akiyama and Urrutia [AU90] study Hamiltonian straight-line alternating paths on equitable point sets in convex positions. They show that it is not always possible to compute a Hamiltonian alternating path on a given equitable point set in convex position and they present an $O(n^2)$-time algorithm that, given an equitable point set in convex position, computes a Hamiltonian alternating path if it exists. Abellanas et al. [AGLHP$^+$99] study the case when points are not restricted to be in convex position. They prove that if either the convex hull of P consists of all the red points and no blue points or if the two point sets are linearly separable, that is, if there exists a straight line that separates the red points from the blue points, then a Hamiltonian straight-line alternating path can always be found. Kaneko et al. [KKS04] study the values of n for which every equitable set of n points admits a Hamiltonian alternating path and proved that this happens only for $n \leq 12$ and $n = 14$. For any other value of n there exist equitable point sets that do not admit a Hamiltonian alternating path. On the other hand, Cibulka et al. [CKM$^+$09] describe arbitrarily large equitable point sets that admit a Hamiltonian straight-line alternating path for every coloring of the points. Non-Hamiltonian alternating paths have also been considered. Abellanas et al. [AGHT03] and Kynčl et al. [KPT08] study the values $\ell(n)$ of the length of a longest straight-line alternating path on sets of red and blue points in convex position and provide upper and lower bounds on $\ell(n)$.

Similar problems have been studied for graph families other than paths. Abellanas et al. [AGLHP$^+$99] investigate *alternating spanning trees*, that is, spanning trees on red and blue point sets such that each edge is a straight-line segment connecting points of different

[1]Throughout the illustrations in this chapter, blue points are black and red points are gray.

colors such that no two edges cross. They prove that every point set $P = R \cup B$ admits an alternating spanning tree whose maximum vertex degree is $O(\frac{|B|}{|R|} + \log |R|)$. Kaneko et al. [KKY00] consider non-planar *Hamiltonian alternating cycles* allowing edge crossings. They prove that at most $n-1$ crossings are sufficient to compute a Hamiltonian alternating cycle and that this is worst-case optimal.

Further, Kano [Kan09] studies general equitable point sets, that is, point sets such that no two points are horizontally and vertically aligned. He shows that general equitable point set admits a perfect matching such that each edge is an L-shaped orthogonal chain connecting a red point to a blue point.

Contribution It is easy to construct equitable point sets that do not allow a Hamiltonian orthogeodesic alternating path on the grid, for instance, a set of horizontally aligned points that are not colored alternatingly red and blue. In contrast to this, we consider general point sets that we already introduced and studied in Chapter 7, that is, point sets whose points are not horizontally or vertically aligned. We show that every general equitable point set with n points admits a Hamiltonian orthogeodesic alternating path and we present an efficient $O(n \log^2 n)$-time algorithm that computes a Hamiltonian orthogeodesic alternating path, given such a point set. The computed path has at most two bends per edge and we prove that this is worst-case optimal. However, the bends along the edges of the computed path may not have integer coordinates. In contrast to this, we show that deciding whether a set of red and blue grid points P admits a Hamiltonian orthogeodesic alternating path with bends at grid points is NP-complete. Further, we describe a $O(n \log^2 n)$-time algorithm that computes an orthogeodesic alternating path of length $|P|/3$ with bends at grid points and we show that there are point sets that do not admit an orthogeodesic alternating path with more than $|P|/2 + 2$ points. Finally, we show that if points of P are allowed to be horizontally or vertically aligned then it is NP-complete to decide whether a balanced point set P has a perfect orthogeodesic alternating matching. This contrasts a recent paper by Kano stating that such a matching always exists if we are not allowed to place more than one point per horizontal or vertical line.

8.2 Hamiltonian Orthogeodesic Alternating Paths

First, we consider the problem of computing a Hamiltonian orthogeodesic alternating path for a given set of red and blue points. Note that there are point sets that do not admit a Hamiltonian orthogeodesic alternating path, for instance, sets of collinear points that are not alternatingly red and blue. In contrast to this, we show that *every* point set with the additional property that no pair of points is horizontally or vertically aligned admits a Hamiltonian orthogeodesic alternating path. The proof is constructive and yields an efficient algorithm to construct a Hamiltonian orthogeodesic alternating path with at most two bends per edge, given a point set with the desired properties. We show that this is also worst-case optimal by showing that there are point sets that do not allow a Hamiltonian orthogeodesic alternating path with at most one bend per edge.

Theorem 8.1. *Every general equitable point set consisting of n red and blue points admits a Hamiltonian orthogeodesic alternating path. Further, a Hamiltonian orthogeodesic path with at most two bends per edge can be computed in $O(n \log^2 n)$ time.*

Before we prove Theorem 8.1, we prove the following auxiliary lemma.

Lemma 8.1. *Let P be a balanced point set with n red and blue points and let $p_1, \dots, p_n$ be the sequence of points sorted from left to right. If $c(p_1) = c(p_n)$, then there is an index i with $2 \leq i \leq n - i$ such that $c(p_i) \neq c(p_1)$ and such that both the point set $P_1 := \{p_1, \dots, p_i\}$ and the point set $P_2 := \{p_{i+1}, \dots, p_n\}$ are non-empty and balanced.*

Proof. Let P be a balanced point set with n red and blue points and let $p_1, \dots, p_n$ be the sequence of points sorted from left to right. Let $P_i := \{p_1, \dots, p_i\}$ and assume without loss of generality that $c(p_1)$ and $c(p_n)$ are both red. Let $r(i)$ and $b(i)$ denote the number of red and blue points in P_i, respectively, and let $f(i) := b(i) - r(i)$. Then $f(1) = -1$ and $f(n-1) = 1$. Further, for $2 \leq i \leq n$, we have $|f(i) - f(i-1)| = 1$ since either $r(i) - r(i-1) = 1$ and $b(i) - b(i-1) = 0$ or $b(i) - b(i-1) = 1$ and $r(i) - r(i-1) = 0$. Since f changes by exactly one unit when going from $i-1$ to i, there must be an index i such that $f(i) = 0$, that is, $r(i) = b(i)$. Hence, P_i is balanced. Since $P = P_n$ is balanced and P_i is balanced, so is $P \setminus P_1 = \{p_{i+1}, \dots, p_n\} =: P_2$. □

Note that, for reasons of symmetry, the lemma also yields that there is an index i such that $c(p_i) \neq c(p_n)$ and such that both $P_1 := \{p_1, \dots, p_{i-1}\}$ and $P_2 := \{p_i, \dots, p_n\}$ are balanced. Now, we prove Theorem 8.1.

Proof of Theorem 8.1. Let P be a general equitable set of n red and blue points such that no pair of points is horizontally or vertically aligned. Let $R \subseteq P$ denote the set of red points and let $B \subseteq P$ denote the set of blue points.

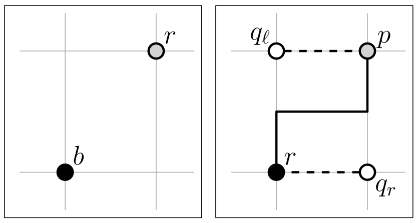
We postpone the unbalanced case and assume that the point set P is balanced. Every Hamiltonian path π on a balanced point set contains one red endpoint r and one blue endpoint b. Thus, we may refer to the ends of the path as the red end and the blue end. If not otherwise stated, we assume that π is directed from its red end to its blue end. First, we show that every balanced point set P admits a Hamiltonian orthogeodesic alternating path π by induction on the size of P such that the following invariants are maintained.

(H1) Let q_ℓ be the point on the left side of the bounding box of P that is horizontally aligned with the red endpoint r and let q_r be the point on the right side of the bounding box of P that is horizontally aligned with the blue end b. Then the straight-line segments $\overline{q_\ell r}$ as well as $\overline{b q_r}$ do not intersect π, except in b and r, respectively, as illustrated in Figure 8.1a.

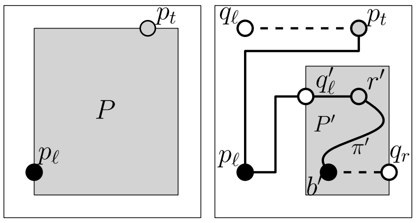
(H2) Each geodesic chain in π has at most two bends.

For convenience, we will assume that the path is directed. Note that, by symmetry, this also shows that we can find a path with the following property.

(H1') Let q_ℓ be the point on the left side of the bounding box of P that is horizontally aligned with the *blue* endpoint b and let q_r be the point on the right side of the bounding box of P that is horizontally aligned with the *red* end r. Then the straight-line segments $\overline{q_\ell b}$ as well as $\overline{r q_r}$ do not intersect π, except in b and r, respectively.



(a) Base case



(b) Case 1.1

Figure 8.1: Illustration for the proof of Theorem 8.1. Base case and Case 1.1.

In the base case of the induction we have $|P| = 2$, that is, P consists of a red point r and a blue point b. We connect r to b by a vertical chain whose horizontal segment is on the line $y = (y(r) + y(b))/2$ as illustrated in Figure 8.1a. Clearly, the invariants are maintained. Now, suppose that the induction hypothesis holds for all balanced point sets with k red and blue points such that $k > 1$ and $2k < n$. Let p_ℓ, p_r, p_t and p_b denote the leftmost, rightmost, topmost and bottommost points on the boundary of the bounding box of P, respectively. Note that some of these points may coincide. We distinguish two cases and several sub-cases as summarized in Figure 8.2.

Case 1: The color of p_ℓ is blue. We distinguish three sub-cases.

Case 1.1: The color of p_t is red. Then $p_t \neq p_\ell$ as illustrated in Figure 8.1b. Let $P' := P \setminus \{p_\ell, p_t\}$. By the induction hypothesis we can compute a path π' on the point set P' such that the invariants *(H1)* and *(H2)* are maintained. That is, π' starts with a red point r' and ends with a blue point b'. Let q'_ℓ denote the point on the left side of $\mathcal{B}(P')$ that is horizontally aligned with r' and let q'_r denote the point on the right side of $\mathcal{B}(P')$ that is horizontally aligned with b'. Further, let p'_t be the topmost point in P' and let p'_ℓ be the leftmost point in P'. We connect p_t to p_ℓ by a vertical chain whose horizontal segment is located on the line $y = (y(p_t) + y(p'_t))/2$

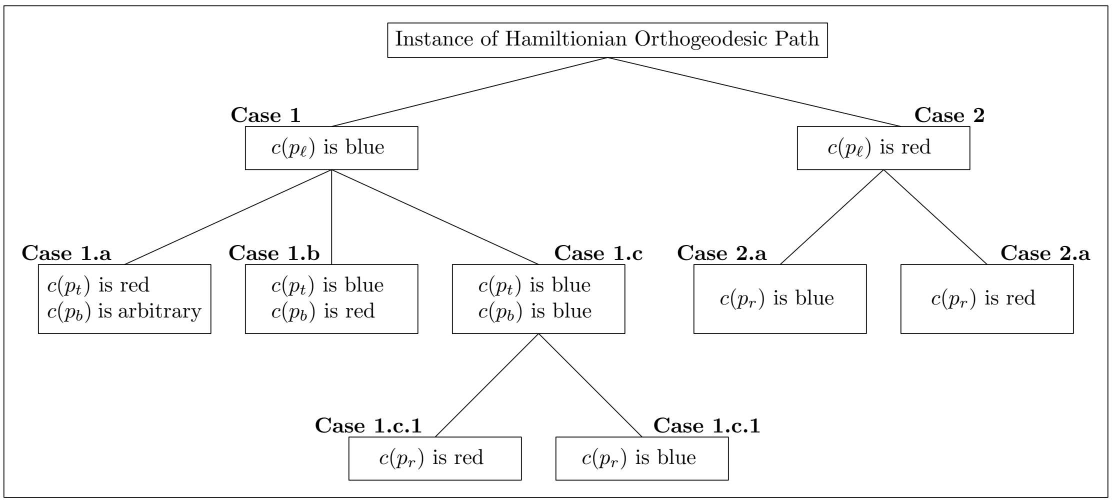
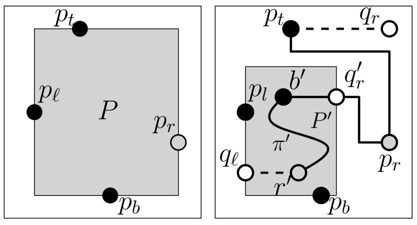


Figure 8.2: Case distinction applied in the proof of Theorem 8.1.


(a) Case 1.3.1

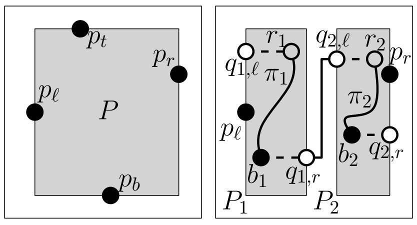

(b) Case 1.3.2

Figure 8.3: Illustration for the proof of Theorem 8.1. Case 1.3.1 and Case 1.3.2.

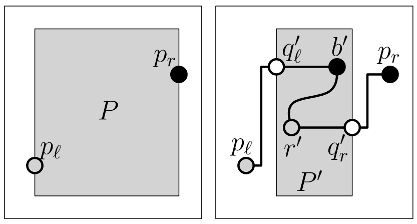
and we connect p_ℓ to r' by a horizontal chain whose vertical segment is on the vertical line $x = (x(p_\ell) + x(p'_\ell))/2$ as illustrated in Figure 8.1b. The resulting path starts with the red point p_t and ends with the blue point b'. Let q_ℓ be the point on the left side of $\mathcal{B}(P)$ that is horizontally aligned with p_t. The point q_r that is horizontally aligned with b' on the right side of $\mathcal{B}(P)$ coincides with q'_r. Then the segments $\overline{q_\ell p_t}$ and $\overline{b' q_r}$ do not intersect π by construction and by the induction hypothesis, respectively.

Case 1.2: The color of p_t is blue and the color of p_b is red. This case is symmetric to Case 1.1 by a reflection at the horizontal axis.

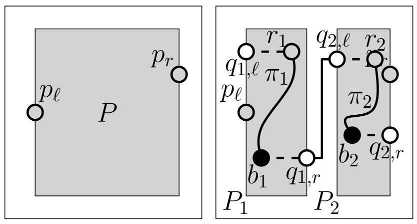
Case 1.3: The color of p_t is blue and the color of p_b is blue. We consider two sub-cases depending on the color of p_r.

Case 1.3.1: The color of p_r is red. This case is similar to Case 1.1. We have $p_r \neq p_t$ since their colors differ. Let $P' := P \setminus \{p_t, p_r\}$. By the induction hypothesis we can compute a path π' on the point set P' such that the invariants *(H1)* and *(H2)* are maintained. That is, π' starts with a red point r' and ends with a blue point b'. Let q'_ℓ denote the point on the left side of $\mathcal{B}(P')$ that is horizontally aligned with r' and let q'_r denote the point on the right side of $\mathcal{B}(P')$ that is horizontally aligned with b'. Further, let p'_t be the topmost point in P' and let p'_r be the rightmost point in P'. We connect p_t to p_r by a vertical chain whose horizontal segment is located on the line $y = (y(p_t) + y(p'_t))/2$ and we connect p_r to b' by a horizontal chain whose vertical segment is on the vertical line $x = (x(p_r) + x(p'_r))/2$ as illustrated in Figure 8.3a. The resulting path π starts with the red point r' and ends with the blue point p_t. Let q_r be the point on the right side of $\mathcal{B}(P)$ that is horizontally aligned with p_t. The point q_ℓ that is horizontally aligned with r' on the left side of $\mathcal{B}(P)$ coincides with q'_ℓ. Then the segments $\overline{q_\ell p_t}$ and $\overline{b' q_r}$ do not intersect π by the induction hypothesis and by construction, respectively.

Case 1.3.2: The color of p_r is blue. According to Lemma 8.1 we can split P into two non-empty balanced point sets P_1 and P_2 that can be separated by a vertical line. By the induction hypothesis, we can compute two Hamiltonian orthogeodesic paths π_1 and π_2 in P_1 and P_2, respectively. Let π_1 be directed from the red point $r_1 \in P_1$ to the blue point $b_1 \in P_1$. Similarly, let π_2 be directed from the red point $p_2 \in P_2$ to the blue point $b_2 \in P_2$. Let $q_{i,\ell}$ and $q_{i,r}$ denote the points on the left and right side of $\mathcal{B}(P_i)$ for $i \in \{1, 2\}$, respectively.



(a) Case 2.1



(b) Case 2.2

Figure 8.4: The different cases of the algorithm. Case 1.2 is a vertical reflection of Case 1.1.

Figure 8.5: Illustration for the proof of Theorem 8.1. Case 2.1 and Case 2.2.

We connect the paths π_1 and π_2 by a horizontal chain between b_1 and r_2 whose vertical segment is on the line centered between $\mathcal{B}(P_1)$ and $\mathcal{B}(P_2)$ as illustrated in Figure 8.3b. The resulting path π is directed from r_1 to b_2. By the induction hypothesis the segments $\overline{q_{1,\ell}r_1}$ and $\overline{b_2q_{2,r}}$ connecting r_1 and b_2 to the left and right side of $\mathcal{B}(P)$, respectively, do not intersect π.

Case 2: The color of p_l is red. We distinguish two sub-cases depending on the color of p_r.

Case 2.1: The color of p_r is blue. Since the size of P is at least four, the point set $P' := P \setminus \{p_\ell, p_r\}$ is non-empty and balanced. By induction hypothesis, we can construct a Hamiltonian orthogeodesic alternating path π' on P' starting at a blue point $b' \in P'$ and ending at a red point $r' \in P'$ such that the following holds.

(H1') Let q'_ℓ be the point on the left side of the bounding box of P that is horizontally aligned with b' and let q'_r be the point on the right side of the bounding box of P that is horizontally aligned with r'. Then the straight-line segments $\overline{q'_\ell b'}$ as well as $\overline{b'q'_r}$ do not intersect π, except in b' and r', respectively.

Let p'_ℓ and p'_r denote the leftmost and rightmost points in P', respectively. Then we connect p_ℓ to b' by a horizontal chain whose vertical segment is on the line $x = (x(p_\ell) + x(p'_\ell))/2$ and we connect r' to p_r by a horizontal chain whose vertical segment is on the line $x = (x(p_r) + x(p'_r))/2$ as illustrated in Figure 8.4a. Since p_ℓ and p_r are on the left and right side of $\mathcal{B}(P)$ the invariant *(H1)* is trivially maintained.

Case 2.2: The color of p_r is red. This case is completely analogous to Case 1.3.2.

Hence, every general balanced point set P admits a Hamiltonian orthogeodesic alternating path. Next, let P be an unbalanced equitable point set and assume without loss of generality that $|R| < |B|$. First, suppose that there is a blue point p on one of the sides of $\mathcal{B}(P)$, say on the left side. Then the point set $P' := P \setminus \{p\}$ is balanced and we can compute a path π' satisfying the invariants *(H1)* and *(H2)* as claimed such that π' starts at a red point r' and ends at a blue point b'. Further, let q'_ℓ be the point on the left side of $\mathcal{B}(P')$ that is horizontally aligned with r'. Then the segment $\overline{q'_\ell r'}$ does not intersect π'. Hence, we can connect p to r' with a horizontal chain whose vertical segment is on the line $x = (x(p) + x(p'_\ell))/2$, where p'_ℓ denotes the leftmost point in P'.

Second, suppose that all points on the boundary of $\mathcal{B}(P)$ are red. Let $r \notin P$ be an arbitrary red point to the left of $\mathcal{B}(P)$ and let $P' := P \cup \{r\}$. Then P' is balanced and both the leftmost point and the rightmost point of P' are red. Then we can split P' into two non-empty point sets P_1 and P_2 that can be separated by a vertical line such that the rightmost point in P_1 is a blue point according to Lemma 8.1. Subsequently, we can compute a path π_1 in P_1 starting at r according to Case 2.1 and we can compute a path π_2 satisfying the invariants *(H1)* and *(H2)* in P_2. The two paths can be concatenated according to Case 2.2 such that the resulting path π is a Hamiltonian orthogeodesic path on P' starting at r. Since r is an end point of π and since we handled the point set according to Case 2.2, we can safely remove r obtaining a Hamiltonian orthogeodesic path on P.

Finally, we show that we can compute a Hamiltonian orthogeodesic path according to the preceding case distinction in $\mathcal{O}(n \log^2 n)$ time. We sort the points with respect to their x- and y-coordinates, respectively, in $\mathcal{O}(n \log n)$ time and we maintain two arrays X and Y containing the points in sorted order. Then each point p can be addressed by two integers $h(p)$ and $v(p)$ denoting the index of p in the horizontal array X and the vertical array Y, respectively. That is we have $X_{h(p)} = p$ and $Y_{h(p)} = p$. Further, we maintain two spatial data structures $\mathcal{R}$ and $\mathcal{B}$ with $O(n \log n)$ initialization time supporting orthogonal range queries in $O(\log n)$ query time [Cha88] for the blue and red points, respectively.

We assume that we are given the bounding box R of the instance P in the form of at most four points p_l, p_r, p_t and p_b on the bounding box of P, each of which is specified by two integers pointing to the position of the points in the horizontal and vertical array, respectively. First we consider all cases, except for Case 1.3.2 and Case 2.2. In these cases we compute the geodesic chain from two points p_1 and p_2 on the boundary of $\mathcal{B}(P)$ and the sub-path computed for $P' := P' \setminus \{p_1, p_2\}$. In order to recurse on P' we need to compute the extremal points of P', given an axis-aligned rectangle $R \supset P'$ with $R \cap P = P \setminus \{p_1, p_2\}$ that can easily be obtained from the extremal points of P' and the horizontal and vertical arrays as follows. Suppose that R is given by two intervals $x \in [i, j]$ and $y \in [k, l]$ where i, j, k and l are integers pointing to the horizontal and vertical arrays, respectively. First we determine the number of points m in $R \cap P$ using the spatial data structures on $\mathcal{O}(\log n)$ time. For each horizontal and vertical side of R that is not yet covered by a point, we perform a binary search on the horizontal and vertical arrays $[i, j]$ and $[k, l]$, respectively, in order to locate the extremal point in P' along the axis orthogonal to the considered side of R. In each iteration of the binary search we query the spatial data structures with the resulting rectangle R' to determine the number of points in R'. If this number is equal to m and the boundary is defined by a point in P', then we have found an extremal point. The test, whether a point p is in P' can be performed by testing whether p is contained in R since we chose R as a rectangle containing exactly the points in P'. Since the number of steps for the binary search is at most $\log n$ and since each step can be performed in $O(\log n)$, we can find the extremal points in $O(\log^2 n)$ time.

In order to split the point set according to Lemma 8.1, note that we can compute the number of red and blue points in a given rectangle in $\mathcal{O}(\log n)$ time using the spatial data structures $\mathcal{R}$ and $\mathcal{B}$. That is, given an index i into the horizontal array of the points, we can compute the function $f(i)$ defined according to Lemma 8.1 in $\mathcal{O}(\log n)$ time. Then we can find an index i splitting the point set with the desired properties by binary search on the indices in $\mathcal{O}(\log^2 n)$ time.

Since each operation of the algorithm can be implemented to run in $O(\log^2 n)$ time and since each type of operation is executed at most n times, the running time of the algorithm is in $O(n \log^2 n)$. □

If P is an equitable point set such that all points have even integer coordinates, then the algorithm computes a Hamiltonian orthogeodesic alternating path such that every bend has integer coordinates, that is, it computes a Hamiltonian path on the grid. Moreover, whenever the horizontal and vertical distance of each pair of points is at least two, we can modify the algorithm to compute a path on the grid as well. Instead of mapping the horizontal and vertical segments of the vertical and horizontal chains onto the bisector of two points as described in the proof of Theorem 8.1, we instead map it to any grid line between the points. Note that, if the horizontal and vertical distance of any pair of points in P is at least two, then there must be at least one unoccupied grid line between the points. Hence, we obtain the following result.

Corollary 8.1. *Let P be an equitable set of n red and blue grid points such that no $\min\{|x(p)-x(q)|, |y(p)-y(q)|\} \geq 2$ for all $p, q \in P$ with $p \neq q$. Then P admits a Hamiltonian orthogeodesic path π with at most two bends per edge such that each bend is located at a grid point. Further, there is an $\mathcal{O}(n \log^2 n)$-time algorithm that constructs such a path.*

The path computed by the algorithm in the proof of Theorem 8.1 has two bends per edge. Next, we show that this is worst-case optimal by showing that there are point sets that do not admit a Hamiltonian orthogeodesic alternating path with one bend per edge.

Theorem 8.2. *For every $n \geq 5$ there exists a general equitable point set that does not admit a Hamiltonian orthogeodesic alternating path with at most one bend per edge.*

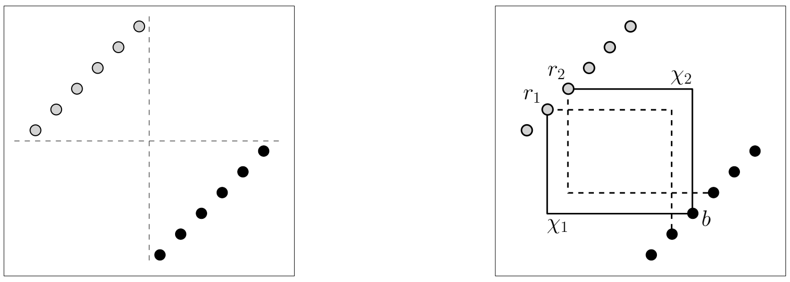
Proof. A *butterfly point set* is an equitable point set P as illustrated in Figure 8.6a with the following properties.

(B1) For every pair of blue points p and q of P, $x(p) < x(q)$ implies $y(p) < y(q)$.

(B2) Similarly, for every pair of red points p and q of P, $x(p) < x(q)$ implies $y(p) < y(q)$.

(B3) For every pair consisting of a blue point p and a red point q of P, $x(p) > x(q)$ and $y(p) < y(q)$. That is, the red points are separable from the blue points by a diagonal line.

Consider a butterfly point set P with at least five points and let π be a Hamiltonian orthogeodesic alternating path on P. Assume without loss of generality that $|R| \leq |B|$. Then every Hamiltonian path on P contains at least one blue point that is not an endpoint of π. Let b be an internal blue point of π. Then b is connected to two red points r_1 and r_2 by two geodesic chains χ_1 and χ_2, respectively, as illustrated in Figure 8.6b. One of the two chains, say χ_1, must have a horizontal segment incident to b while the other chain, that is χ_2, must have a vertical segment incident to b. If b is connected to r_1 and r_2 by orthogeodesic chains with at most one bend, then χ_1 must be attached to r_1 by a vertical segment, while χ_2 must be attached to r_2 by a horizontal segment. Since all red points are above and to the left of all the blue points and since the two chains cannot cross, we have $x(r_1) < x(r_2)$. At least one of the points r_1 and r_2 must be connected to a blue point b' distinct from b since π can have at most one red point as an endpoint due to $|R| \leq |B|$. If $x(b') < x(b)$ then the geodesic



(a) (b)

Figure 8.6: (a) A butterfly point set. (b) Illustration for the proof of Theorem 8.2. If point p is connected to r_1 and r_2 with two 1-bend geodesic chains, then r_1 and r_2 cannot be connected to any other blue point by a geodesic chain without introducing a crossing.

chain connecting r_1 or r_2 to b' would cross chain χ_1. If $x(b') > x(b)$ then the geodesic chain connecting r_1 or r_2 to b' would cross chain χ_2 as illustrated in Figure 8.6b. Thus, two bends are worst-case optimal.

□

8.3 Hamiltonian Orthogeodesic Alternating Paths on the Grid

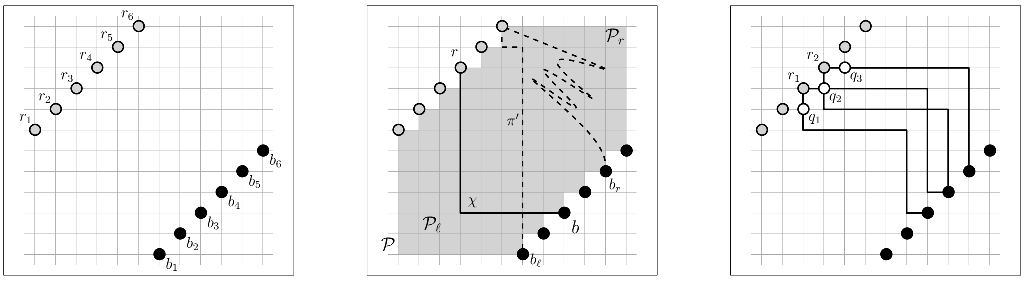
While we have seen that we can always construct a Hamiltonian orthogeodesic alternating path on the grid if the horizontal and vertical distance between any pair of points is at least two, a Hamiltonian orthogeodesic path does not always exist if we drop this requirement and consider point sets whose points are neither horizontally not vertically aligned.

Theorem 8.3. *For every $n \geq 5$, there exists a general equitable set P of n red and blue grid points that does not admit a Hamiltonian orthogeodesic alternating path on the grid.*

Proof. Let P be a butterfly point set with at least five points with the additional property that the points have integer coordinates. Let $R := \{r_1, \dots, r_{|R|}\}$ and $B := \{b_1, \dots, b_{|B|}\}$ denote the red and blue points and let P be such that $|R| \leq |B|$ and such that $x(r_i) = i$, $y(r_i) = i$ for all $1 \leq i \leq |R|$ as well as $x(b_i) = |R| + i$ and $y(b_i) = i - |B|$ for $1 \leq i \leq |B|$, respectively, as illustrated in Figure 8.7a.

First, note that, since each orthogeodesic chain is contained inside the bounding box of its endpoints and since the endpoints have different color, π must be contained in the polygon $\mathcal{P}$ defined by the union of the rectangles spanned by all pairs of red and blue points, respectively, as illustrated in Figure 8.7b.

Next, we show that the endpoints of π must be leftmost or rightmost red or blue points, respectively. Suppose, for instance, that π starts in a blue point b such that there is a blue point b_ℓ to the left and a blue point b_r to the right of b, respectively. Let π' denote the sub-path of π ending at b_ℓ and b_r, respectively. Further, let r be the unique red point adjacent to b and let χ be the orthogeodesic chain connecting b and r. Clearly χ bisects $\mathcal{P}$ into two sub-polygons $\mathcal{P}_\ell$ and $\mathcal{P}_r$ such that b_ℓ is contained in $\mathcal{P}_\ell$ and b_r is contained in $\mathcal{P}_r$. That is,



(a) (b) (c)

Figure 8.7: Illustration for the proof of Theorem 8.3. (a) Butterfly one the grid. (b) If a path starts neither in blue point that is neither the leftmost nor the rightmost blue point, then there must be a crossing. (c) Two internal red points of the path cannot be next to each other on the grid, since the four incident orthogeodesic chains can use only three distinct points incident to the red points.

the path π' must cross χ as illustrated in Figure 8.7b. Hence, the endpoints of π must be leftmost or rightmost red or blue points, respectively, as claimed.

Finally, we claim that π contains at least one pair of internal red points r_1 and r_2 such that $x(r_2) = x(r_1) + 1$. If n is odd, π has two blue points as its endpoints since we assumed $|R| \leq |B|$. Hence we can find two red points with the desired properties. Otherwise, if n is even we have $n \geq 6$ and π has both a red and a blue point as its endpoints. Since we argued that the endpoints must be leftmost or rightmost red and blue points, respectively, we can find a pair of red points with the desired properties since the red endpoint of π must be the leftmost or rightmost point of the at least three red points.

Let r_1 and r_2 be two internal red vertices of π such that $x(r_2) = x(r_1) + 1$, that is, $y(r_2) = y(r_1) + 1$. Since r_1 and r_2 are internal points of π, both r_1 and r_2 are adjacent to two orthogeodesic chains each. Since all blue points are right of and below r_1 and r_2, respectively, these chains must occupy the horizontal and vertical grid lines starting at r_1 and r_2 in the downward and rightward direction, respectively. That is, these chains must occupy the grid points one unit to the right and one unit below r_1 and r_2, respectively. However, this implies that the four distinct orthogeodesic chains incident to p_1 and p_2 must occupy three distinct points $q_1 := (x(r_1), y(r_1) - 1)$,$q_2 := (x(r_1) + 1, y(r_1)) = (x(r_2), y(r_2) - 1)$ and $q_3 := (x(r_2) + 1, y(r_2))$. Thus, at least two of the chains must intersect as illustrated in Figure 8.7c.

Note however, that any butterfly point set with the desired properties on the grid with at most four vertices does admit a Hamiltonian orthogeodesic alternating path on the grid since such a path never contains two internal red or blue points, respectively. □

Motivated by Theorem 8.3, we study the Hamiltonian Orthogeodesic Alternating Path on the Grid problem, that is, the problem of deciding whether a given general equitable set of grid points admits a Hamiltonian orthogeodesic alternating path on the grid. Surprisingly it turns out that this problem is NP-complete. If we are allowed to place more than one point on a horizontal or vertical line, we can show that it is even NP-complete to decide whether there exists an orthogeodesic alternating perfect matching. This contrasts a result by Kano [Kan09] stating that such a matching always exists if we are not allowed to place more than one point per horizontal or vertical line.

Theorem 8.4. Hamiltonian Orthogeodesic Alternating Path on the Grid *is NP-complete.*

First, we show containment of Hamiltonian Orthogeodesic Alternating Path on the Grid in $\mathcal{NP}$.

Lemma 8.2. Hamiltonian Orthogeodesic Alternating Path on the Grid *is contained in* $\mathcal{NP}$.

Proof. In order to show that Hamiltonian Orthogeodesic Alternating Path on the Grid is contained in NP, we introduce the notion of a *bottommost* orthogeodesic path and show that the problem can be reduced to deciding whether there is a bottommost Hamiltonian orthogeodesic alternating path for P. An instance of this problem can be encoded in polynomial space and verified in polynomial time.

We use the terminology introduced in Section 9. We say that a point $p \in \mathbb{R}^2$ k-dominates the points in k-th quadrant of the orthogonal coordinate system with origin at p. The unbounded range corresponding to the k-the quadrant is called k-cone. Given a set Q of points, we refer to the union of the k-cones of the points in Q as the *orthogeodesic k-hull of Q* as illustrated in Figure 8.8a. Further, by $Q^{\nwarrow}$ we denote the set of points resulting from translating the points in Q one unit to the left and one unit to the top and by $Q^{\nearrow}$ we denote the set of points resulting from translating the points in Q one unit to the right and one unit to the top.

Assume we are given an orthogeodesic path $\pi = (P, E)$ on a point set P such that each $e \in E$ is an orthogeodesic chain. We denote the leftmost point of e by e^- and its rightmost point by e^+. An edge is called *upward* if $y(e^+) \geq y(e^-)$, otherwise, it is called *downward*. We define the partial order $\prec$ on E such that for $e_1, e_2 \in E$ we have $e_1 \prec e_2$ if and only if there is a vertical line intersecting e_1 below e_2. The path π is called *bottommost orthogeodesic path* if and only if each edge e is embedded as the bottommost orthogeodesic chain with respect to $\prec$. By this we mean, that each upward edge e connecting p and q is embedded on the orthogeodesic 4-hull of the point set $\{p, q\} \cup P_e^{\nwarrow} \cup B^{\nwarrow}$, and each downward edge is embedded on the orthogeodesic 2-hull of the point set $\{p, q\} \cup P_e^{\nearrow} \cup B^{\nwarrow}$, where P_e is the union of the endpoints of all edges that are smaller than e with respect to $\prec$ and B is the set of bends induced by the bottommost chains of these edges. That is, each smallest edge with respect to $\prec$ is embedded as an L-shaped chain consisting of one horizontal and one vertical straight-line segment.

Suppose that π is a bottommost orthogeodesic path. Then it is easy to see that the number of bends of each edge is bounded by a linear function in the number of points as follows. First, note that each of the bends corresponding to a 2-cone or a 4-cone, respectively, is obtained by translating a point of the original point set. Further, the total number of bends is at most twice this number of bends $+1$. Hence, all bends of π are placed on a polynomial number of points $\widehat{P}$ such that

$$\widehat{P} := \{(x+i, y+i), (x+i, y-i) \mid (x, y) \in P \wedge 1 \leq i \leq n-1\}$$

since π has $n-1$ edges.

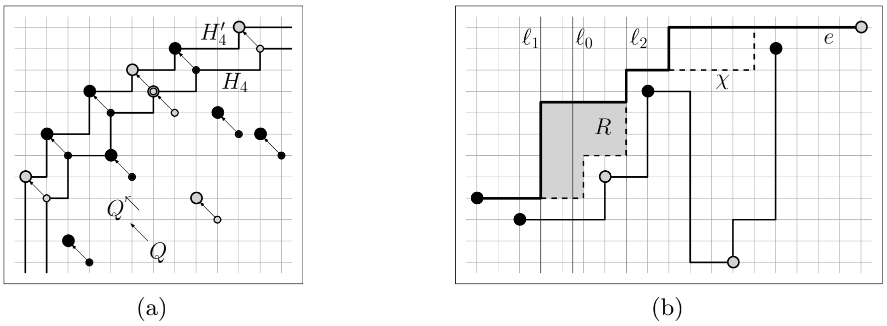


Figure 8.8: Illustrations for the proof of Lemma 8.2. (a) Point set Q (small points) and 4-hull H_4 of Q as well as $Q^{\nwarrow}$ (large points) and 4-hull H_4' of $Q^{\nwarrow}$. (b) An edge e of a Hamiltonian orthogeodesic alternating path that is not embedded as the bottommost geodesic chain χ.

Further, it is clear that every path can be transformed into a bottommost path without changing the partial order $\prec$ as follows. Suppose that $\pi = (P, E)$ is a Hamiltonian orthogeodesic alternating path and let $\prec$ be the partial order on the edges of E defined as above, that is $e' \prec e$ if and only there is a vertical line that intersects e' below e for $e', e \in E$. Assume that $e \in E$ an upward edge that is not embedded as the bottommost orthogeodesic chain with respect to $\prec$ but all edges $e' \in E$ with $e' \prec e$ are embedded as the bottommost orthogeodesic chain. The case, when e is a downward edge can be handled similarly.

Let χ be the bottommost orthogeodesic chain between e^- and e^+ with respect to $\prec$. We claim that every vertical line intersecting e intersects χ below e or it intersects both chains in the same point. Suppose that there is a vertical line ℓ such that ℓ intersects χ above e. Since e is embedded according to $\prec$, it must be embedded above or on the orthogeodesic hull of the points $\{e^-, e^+\} \cup P_e^{\nwarrow}$ where P_e denotes the union of the endpoints of all edges that are smaller than e with respect to $\prec$. Otherwise, we would find an edge $e' \prec e$ and a vertical line that intersects e' above e or that intersects both e' and e in the same point. On the other hand, if at least one bend b of e is embedded in the region that is 4-dominated by the point set B of all bends corresponding to edges that are smaller than e with respect to $\prec$, then we again find a vertical line intersecting e below some edge e' with $e' \prec e$. Hence e must be embedded above or on χ defined as the orthogeodesic 4-hull of $\{p, q\} \cup P_e^{\nwarrow} \cup B^{\nwarrow}$. Hence every vertical line intersecting e either intersects χ below e or it intersects both chains in the same point.

Since $e \neq \chi$ we can find a vertical line ℓ_0 such that ℓ_0 intersects e above χ as illustrated in Figure 8.8b. Let ℓ_2 be the leftmost line to the right of ℓ_0 such that ℓ_2 intersects e and χ in the same point. Further, let ℓ_1 be the rightmost vertical line left of ℓ_0 such that ℓ_1 intersects both e and χ in the same point. Note that ℓ_1 and ℓ_2 are well-defined since the vertical lines through e^- and e^+ have the desired property, respectively. Further, let R be the region enclosed between ℓ_1, ℓ_2, χ and e as illustrated in Figure 8.8b. We claim that R is empty. Note that each edge intersecting the vertical strip between ℓ_1 and ℓ_2 is comparable to e with respect to $\prec$. Then R does not contain any edge e' with $e \prec e'$ by the definition of $\prec$. On the other hand, all edges e' with $e' \prec e$ are below χ which is below e. Hence, R is empty and we can substitute e by a new orthogeodesic chain consisting of the first part of e between e^- and ℓ, the last part of e between ℓ_r and e^+ as well as an intermediate part consisting of the sub-chain of χ between ℓ_1 and ℓ_2 and the vertical segment of ℓ_2 between χ and e. By iteratively applying this argument to any smallest edge with respect to $\prec$ that is not embedded as the bottommost orthogeodesic chain, we can iteratively contract the space between e and χ such that e will eventually be embedded as the bottommost orthogeodesic chain.

Thus, the problem of deciding the Hamiltonian Orthogeodesic Alternating Path on the Grid problem is equivalent to deciding whether there is a bottommost Hamiltonian orthogeodesic alternating path on P. Such a path can be uniquely encoded by the sequence of points along the path and the partial order $\prec$. Given this, we can check if the uniquely determined bottommost path is a Hamiltonian orthogeodesic path in polynomial time by computing the path and checking planarity. Hence, the problem is in NP. $\square$

Next, we prove that Hamiltonian Orthogeodesic Alternating Path on the Grid is NP-complete.

Proof of Theorem 8.4. The problem is in NP as stated in the previous Lemma 8.2. We show NP-hardness by reduction from 3-Partition using similar techniques as in the proof of Theorem 6.4. Recall that an instance of 3-Partition consists of a multiset $A = \{a_1, \dots, a_{3m}\}$ of $3m$ positive integers, each in the range $(B/4, B/2)$, where $B = (\sum_{i=1}^{3m} a_i)/m$, and the question is whether there exists a partition of A into m subsets $A_1, \dots, A_m$ of A, each of cardinality 3, such that the sum of the numbers in each subset is B. Since 3-Partition is *strongly* NP-hard [GJ79], we may assume that B is bounded by a polynomial in m.

Given an instance A of 3-Partition, we construct a corresponding instance $P = R \cup B$ of the Hamiltonian Orthogeodesic Alternating Path on the Grid problem such that P allows for a Hamiltonian orthogeodesic alternating path if and only if there exists a partition of A with the desired properties as follows.

A sequence $p_1, \dots, p_k$ of diagonally aligned grid points is called *k-spaced* if the Euclidean distance between subsequent points p_i and p_{i+1} is exactly $k\sqrt{2}$ for all $1 \le i \le k-1$. The point set P of constructed instance consists of four different types of points, called *hinge points*, *element points*, *mask points* and *partition points*, and is aligned on a regular sawtooth-pattern with $3m+2$ teeth, numbered $T_0, \dots, T_{3m+1}$ from left to right. The point set, as well as the sawtooth-pattern and the teeth are illustrated in Figure 8.9.

Let L be some integer to be specified later. Each tooth T_i consists of a diagonal segment with slope 1 of length $L\sqrt{2}$, denoted by S_i, and a diagonal segment with slope -1 of length $(2L+1)\sqrt{2}$. Hence, the tips of the teeth are aligned along a line with negative slope such that the tip of T_i is below the lowest point of S_{i-1} for $1 \le i \le 3m+1$. We construct our point set as follows.

Along S_0, we align $m^2B + 4m$ 2-spaced blue hinge points starting at the leftmost point of S_0. For each element a_i we align $2a_i + 1$ 1-spaced red element points along S_i. Further, we align m sets of B 2-spaced blue partition points along S_{3m+1}, each acting as a partition. These partitions are separated by $m-1$ sequences of $mB+1$ 2-spaced red mask points which will act as a sort of "dot mask" separating partitions that are consecutive along S_{3m+1}. The maximal sequences of blue points along S_{3m+1} are called *partitions* and the maximal sequences of red points along S_{3m+1} are called *masks*. By construction P contains $m^2B + mB + 4m - 1$ red and $m^2B + mB + 4m$ blue points and, thus, is equitable with one more blue point. Hence,

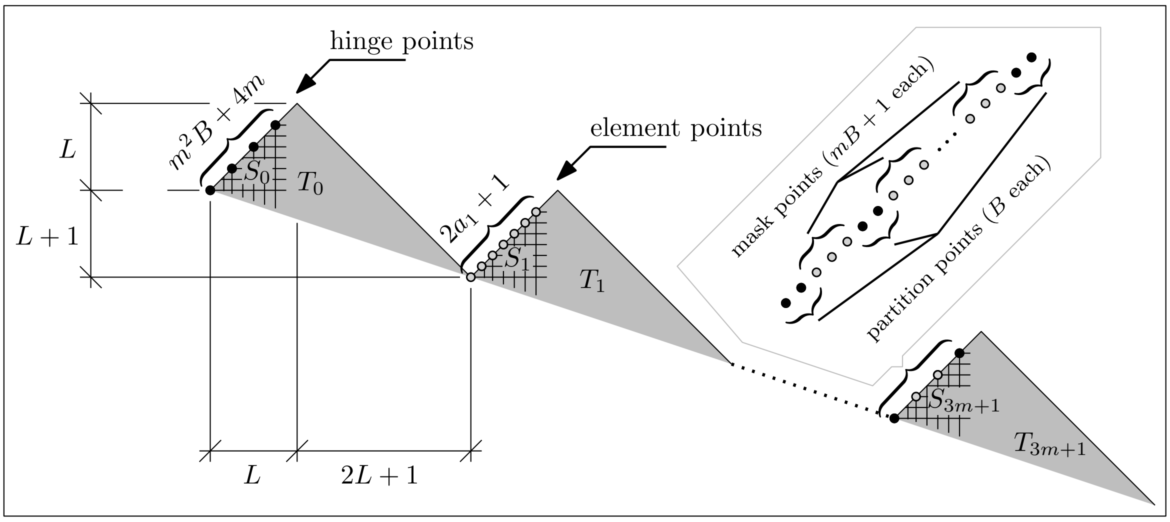


Figure 8.9: Pointset used in the reduction. Each shaded triangle constitutes a tooth T_i. All points are arranged on the ascending slope S_i of T_i.

any alternating path must start and end with a blue point and all red points must be interior points of the path. This implies that every red point must be connected to exactly two blue points.

We now show that there is a partition of A with the desired properties if and only if the point set contains a Hamiltonian orthogeodesic alternating path. A high-level illustration of the reduction is given in Figure 8.10. Assume that we are given a Hamiltonian orthogeodesic alternating path π on P. First, consider the red mask points. In each mask there must be one mask point that is connected to a blue hinge point on S_0. To see this, note that there are $mB + 1$ red mask points in each of the $m - 1$ masks, each of which is adjacent to two blue points on π. Further, each blue point can have at most two adjacent red points on π. Hence, the red masks points of a fixed mask are adjacent to a total of at least $mB + 1$ blue points. Since there are only mB blue points in total on S_{3m+1}, the red mask points of each of the masks must be adjacent to at least one blue hinge point each. Each edge between a red mask point and a blue hinge point is called a *partitioner*. Consider the partial order $\prec$ on the edges of π such that $e' \prec e$ if there is a vertical line intersecting e' below e. Since there is a vertical line ℓ_0 that is intersected by all partitioners, the partitioners are totally ordered by $\prec$. Let the sequence of partitioners sorted with respect to $\prec$ be given by $\mathcal{P}_1, \dots, \mathcal{P}_k$. Note that $k \geq m - 1$. For convenience we extend the partitioners by imaginary horizontal lines towards the left and right. Then the k partitioners partition the plane into $k + 1$ regions $R_1, \dots, R_k + 1$ such that R_1 is the region below $\mathcal{P}_1$, R_i is the region bounded by $\mathcal{P}_{i-1}$ and $\mathcal{P}_i$ and R_{k+1} is the region above $\mathcal{P}_{k+1}$.

Next, consider the element points. Since the element points corresponding to a single element are 1-spaced, no partitioner can pass between them on a grid line. Hence, the partitioners will partition the element points according to the element sizes, such that all element points corresponding to a single element are contained in the same partition. Each element a_i can then be associated with a unique index $f(i)$ such that all element points corresponding to a_i are contained in $R_{f(i)}$. However, we still need to show that each partition

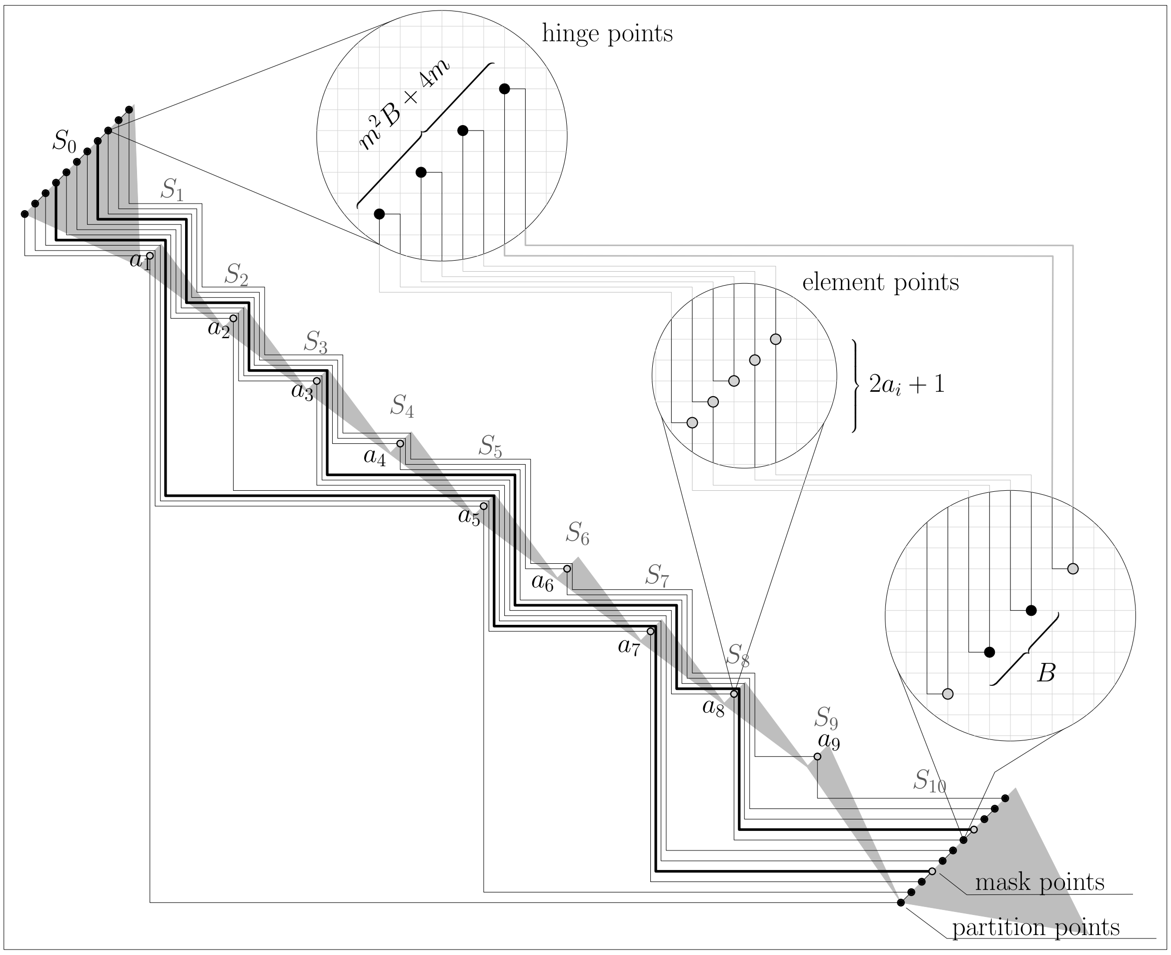


Figure 8.10: A high-level illustration of an exemplary reduction from 3-Partition to Hamiltonian Orthogeodesic Alternating Path on the Grid using the instance $A_1 = \{a_1, a_5, a_7\}$, $A_2 = \{a_2, a_3, a_8\}$, $A_3 = \{a_4, a_6, a_9\}$ (not to scale). Details are depicted in the circles.

contains the correct number of element points.

Let D_i be the diagonal line through S_i and let H_i^+ and H_i^- denote the upper and lower half-planes defined by D_i, respectively. We claim that each group of $2a_i + 1$ element points corresponding to element a_i can have at most $2a_i + 2$ blue incidences in H_i^+. Each of these incidences is a geodesic chain starting either with a horizontal segment to the left or with a vertical segment towards the top. These segments can be covered by gridpoints adjacent to the element points. As there are only $2a_i + 2$ such gridpoints, the claim holds.

Recall that the element points must be interior points of the path since P contains more blue points than red points, that is, each red element point must be adjacent to two blue points in π. Since the group of element points corresponding to a_i must therefore have $4a_i + 2$ blue incidences in total, and since it can have at most $2a_i + 2$ blue incidence in H^+, it must have at least $2a_i$ blue incidences in H_i^-. Thus, the union of all red element points must have a total of $2mB$ blue incidences on S_{3m+1}. On the other hand, there are only mB blue points on S_{3m+1}, each of which must have two red incidences. This implies that element a_i has exactly $2a_i$ incidences in H_i^- and that the blue partition points are connected only to the element points.

Finally, consider the partitions. Clearly the B partition points corresponding to a fixed partition on S_{3m+1} must all be contained in the same region R_j. Otherwise, some partitioner, say $\mathcal{P}_s$ must pass between two partition points. However, such a partitioner would have to connect to a red mask point above or to the left of it, both resulting in an orthogonal chain that would not be orthogeodesic. Since there are B blue points in each of the partitions, the number of element points must add up to $2B$, that is, the corresponding elements add up to B and thus yield a valid 3-partition of A.

Conversely, suppose that we are given a valid partition $A_1, \ldots, A_m$ of A according to 3-Partition. Then we can find a Hamiltonian orthogeodesic alternating path as follows. We iteratively embed the geodesic chains such that each geodesic chain is drawn as the bottommost geodesic chain as defined in the proof of Lemma 8.2 that runs one grid unit above all geodesic chains embedded so far as illustrated in Figure 8.10.

We start with an arbitrary partition, say A_1 containing elements a_i, a_j, a_k such that $i < j < k$. First, we consider the element a_i. We draw an alternating path starting at the leftmost hinge point using the first a_i partition points, the first $2a_i + 1$ element-points corresponding to a_i as well as the leftmost $a_i + 2$ hinge points on S_0. The path alternates between the hinge points and the partition points, visiting the element points in between and ends at a blue hinge point. We proceed accordingly for elements a_j and a_k, respectively, in this order. Next, we embed a sequence of edges corresponding to partitioners. We start with the blue hinge point that we ended after visiting the last element point of a_k and we alternatingly visit consecutive blue hinge points and red mask points, ending again, at a blue hinge point. The remaining elements are handled in an analogous manner. Since each edge is embedded as the bottommost orthogeodesic chain, it is below all points to be inserted in later iterations. Further, let the parameter L used earlier on in the construction be defined as $L := 2m^2B + 2mB + 8m - 1$, that is, L is equal to the number of points of P. This implies that there are at least $L - 1$ unoccupied grid lines between any pair of element points corresponding to different elements. Since the constructed path has $L - 1$ edges, we therefore did not introduce any crossings. Hence, we have constructed a Hamiltonian orthogeodesic alternating path on P. □

Note that we may add another red point above and to the left of all points in P to make the point set balanced. Using arguments analogous to the arguments used in the proof of Theorem 8.4 we can show the following corollary.

Corollary 8.2. *It is NP-complete to decide whether a given balanced set of red and blue grid points such that no two points are on a common horizontal or vertical line allows for a Hamiltonian orthogeodesic alternating cycle if bends are only allowed at grid points.*

Kano [Kan09] showed that every balanced set of red and blue points such that no two points are on a common horizontal or vertical line admits a perfect orthogeodesic alternating matching consisting of L-shaped orthogonal chains. Hence such a matching is completely on the grid whenever the points are grid points. Surprisingly, the problem becomes NP-complete if the points are allowed to be horizontally and vertically aligned. The proof for the following theorem is similar to the proof of Theorem 8.4.

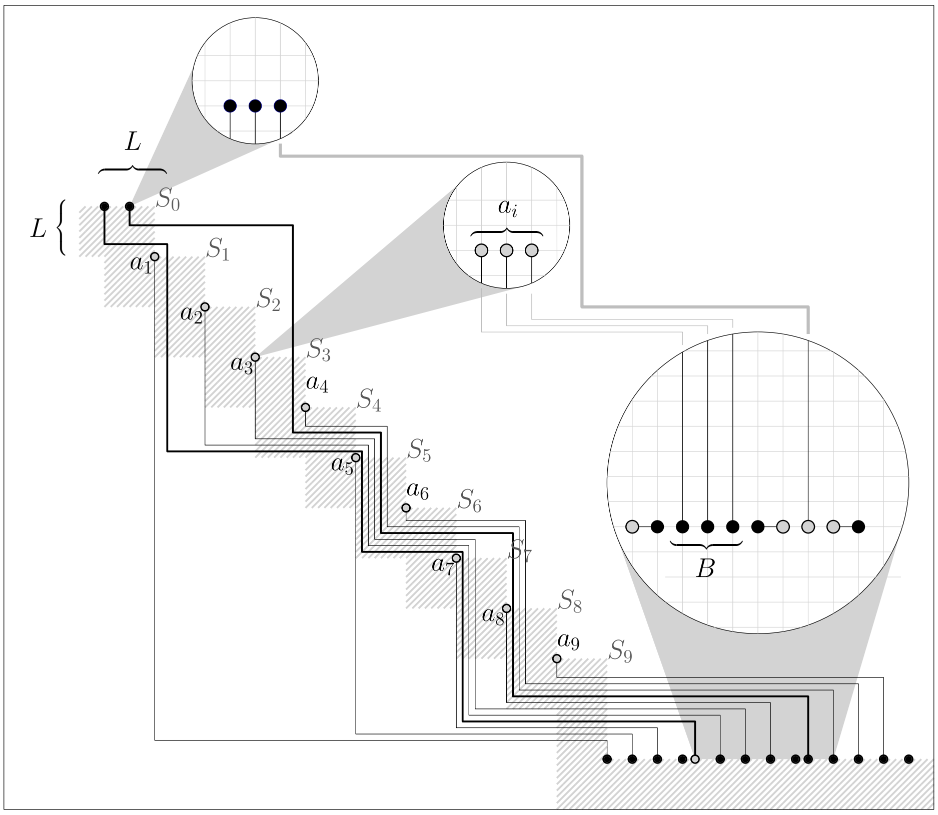


Figure 8.11: Example of the reduction from 3-Partition to Perfect Orthogeodesic Alternating Matching on Grid using $A_1 = \{a_1, a_5, a_7\}$, $A_2 = \{a_2, a_3, a_8\}$, and $A_3 = \{a_4, a_6, a_9\}$ (not to scale)

Theorem 8.5. *Given an arbitrary balanced set of red and blue grid points, it is NP-complete to decide whether there is a perfect orthogeodesic alternating matching on the grid.*

Proof. Showing containment in NP is analogous to the proof of Lemma 8.2. We show that the problem is NP-hard by reduction from 3-Partition similar to the proof of Theorem 8.4. Given an instance A of 3-Partition, we construct a corresponding instance $P = R \cup B$ of the Perfect Orthogeodesic Alternating Matching on Grid problem as illustrated in Figure 8.11 such that P allows for a perfect orthogeodesic alternating matching if and only if there exists a partition of A with the desired properties.

A set of horizontally aligned grid points is called *k-spaced* if the Euclidean distance between two adjacent points is exactly k. As in the proof of Theorem 8.4, the point set P consists of four different types of points, called *hinge points*, *element points*, *mask points* and *partition points*, yet now the point set is aligned a regular staircase on the grid with $3m + 1$ stairs such that each stair has width and height $L := \lceil B/2 \rceil + 3m$. We number the stairs $S_0, \ldots, S_{3m}$ starting at the top. Then we construct our point set as follows.

On the horizontal line of the topmost stair S_0, we align $m - 1$ 1-spaced blue hinge points starting at the leftmost point of S_0. For each element a_i we align a_i 1-spaced red element points along the horizontal stair S_i. On the bottom line of the staircase, we align m sets of $B + 2$ 1-spaced blue partition points, each acting as a partition. These partitions are each separated by three 1-spaced red mask points that are placed at distance 1 from the partitions and which will act as a sort of “dot mask” separating the partitions.

Clearly, the instance is balanced. Since we consider matchings, each red point must be connected to exactly one blue point. We show that there is a perfect alternating matching if and only if there is a partition of A with the desired properties. Assume we are given a perfect orthogeodesic alternating matching.

First, consider the red mask points in the middle of each mask. Each of those red mask points can only be connected to one of the hinge points since it is flanked by red points on both sides. Due to the horizontal alignment of both the mask points and the hinge points, these incidences are uniquely determined. We call any geodesic chain connecting a mask point and a hinge point a *partitioner.* All the remaining red mask points must connect to the unique adjacent blue point on the bottommost stair since all other blue points are already matched to the red mask points in the middle.

Since the element points corresponding to a single element are 1-spaced, no partitioner may pass between them and, hence, the partitioners partition the elements such that the element points corresponding to a single element are all in the same partition.

Now consider the red element points. Each of these points must be connected to a blue partition point, since these points are the only remaining blue points. Since the elements have been partitioned by the partitioners and since there are exactly B blue points in each partition, it is clear that the existence of the matching implies the existence of a partition of A, which is obtained from the matching in a straightforward manner.

Conversely, given a partition, we can easily construct a valid perfect orthogeodesic alternating matching. As in the proof of Theorem 8.4, each geodesic chain is drawn as the bottommost geodesic that runs above all geodesics drawn so far. We start with the element a_i with the smallest index in A_1. We connect the element points of a_i to the leftmost a_i blue partition points that have not yet been used. Then we proceed in the same manner with the second and third element from the first partition. After that we draw the partitioner connecting the leftmost hinge point with the leftmost middle mask point. We proceed accordingly with the remaining partitions. Due to the space we reserved between the elements along the staircase, it is clear, that we can draw all geodesic chains in the desired way. □

8.4 Long Orthogeodesic Alternating Paths on the Grid

Motivated by the hardness of deciding whether a given equitable set of red and blue points admits a Hamiltonian orthogeodesic alternating path on the grid according to Theorem 8.4, we consider the following optimization problem. Given an equitable set of red and blue grid points P such that no two points are on a common horizontal or vertical line, we wish to find a subset $P' \subseteq P$ of maximum size such that P' admits a Hamiltonian orthogeodesic alternating path on the grid. First we show that there are point sets consisting of $2n$ points that do not admit a Hamiltonian orthogeodesic alternating path of length more than $n + 1$.

Theorem 8.6. *For every $n \geq 6$ there exists a general equitable point set P consisting of n red and blue points such that the largest point set $P' \subseteq P$ admitting a Hamiltonian orthogeodesic path on the grid has at most $\lfloor n/2 \rfloor + 2$ points.*

Proof. Consider a butterfly point set P consisting of n red and blue points on the grid. Let $R := \{r_1, \dots, r_{|R|}\}$ and $B := \{b_1, \dots, b_{|B|}\}$ denote the red and blue points of P and let P

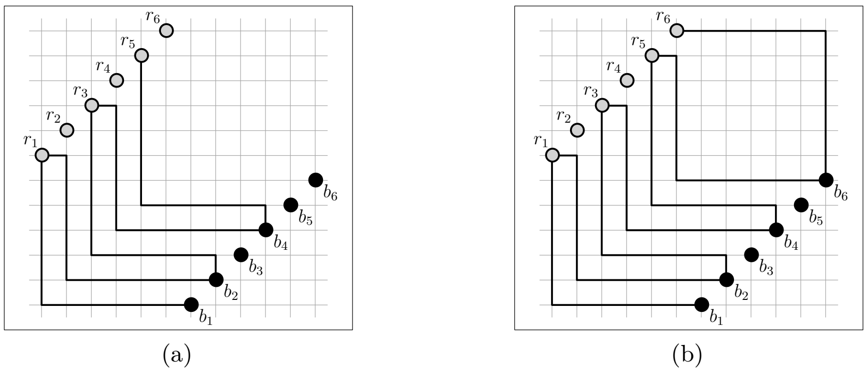


Figure 8.12: Illustration for the proof of Theorem 8.6 depicting a butterfly point set on the grid and a longest orthogeodesic alternating path.

be such that $|R| \leq |B|$ and such that $x(r_i) = i$, $y(r_i) = i$ for all $1 \leq i \leq |R|$ as well as $x(b_i) = |R| + i$ and $y(b_i) = i - |B|$ for $1 \leq i \leq |B|$, respectively, as illustrated in Figure 8.7a. Further, let π be an orthogeodesic path with maximum length on a point set $P' \subseteq P$. Since the points are not horizontally and vertically aligned, each edge consists of at least two straight-line segments. Thus, each straight-line segment of π is incident to at most one point in P'. Let S denote the set of straight-line segments of π that are incident to a point in P' and let each segment in S be colored according to the unique point to which it is incident. That is, S contains $|P'| - 1$ red and $|P'| - 1$ blue segments. Each of the segments in S covers a grid point adjacent to the unique point in P' to which it is incident. Since all blue points are to the right and below all red points, each orthogeodesic chain incident to a red point p covers a grid point one unit to the right or one unit below r. Similarly, each orthogeodesic chain incident to a blue point b covers a grid point one unit to the left or one unit above b. Thus, the red straight-line segments of π cover a total of $|R| + 1$ distinct grid points adjacent to the red grid points and the blue straight-line segments of π cover a total of $|B| + 1$ distinct points adjacent to the blue grid points. Since $|R| \leq |B|$, there are at most $|R| + 1$ red segments, that is, π contains at most $|R| + 1$ edges and at most $|R| + 2$ points. Clearly, $\lfloor n/2 \rfloor + 2 \leq |R| + 2$ follows from $|R| + |B| = n$ and $|B| \leq |R| + 1$. Hence, there cannot be an orthogeodesic alternating path containing more than $\lfloor n/2 \rfloor + 2$ points.

Next we show that this is tight by proving that we can construct an orthogeodesic path of this length on P. Let $f(i)$ be defined such that

$$f(i) := \begin{cases} \frac{i}{2} - 1 & \text{if } i \text{ is even} \\ \frac{i-1}{2} & \text{if } i \text{ is odd} \end{cases} .$$

We start by connecting b_1 to r_1 by an L-shaped orthogeodesic chain consisting of a horizontal segment incident to b_1 and a vertical segment incident to r_1. For $1 \leq i \leq f(|R|)$ we connect r_{2i-1} to b_{2_i} by a horizontal chain whose vertical segment is one unit to the right of r_{2i-1} and we connect b_{2i} to r_{2i+1} by a vertical chain whose horizontal segment is one unit above b_{2i} as illustrated in Figure 8.12a.

Case 1: $|R|$ is odd. Then the constructed path has $2f(|R|) + 1 = |R|$ edges and ends in $r_{2f(|R|)+1} = r_{|R|}$. We connect $r_{|R|}$ to $b_{|R|}$ by an L-shaped edge composed of a horizontal segment incident to $r_{|R|}$ and a vertical segment incident to $b_{|R|}$, which yields an orthogeodesic path on $|R| + 2 \geq \lfloor n/2 \rfloor + 2$ points in total.

Case 2: $|R|$ is even. Then the constructed path has $2f(|R|) + 1 = |R| - 1$ edges and ends in $r_{2f(|R|)+1} = r_{|R|-1}$ as illustrated in Figure 8.12a. We connect $r_{|R|-1}$ to $b_{|R|}$ by a horizontal chain whose vertical segment is one unit to the right of $r_{|R|-1}$. Further, we connect $b_{|R|}$ to $r_{|R}$ by an L-shaped orthogeodesic chain that is vertically attached to $b_{|R|}$ and horizontally attached to $r_{|R|}$ as illustrated in Figure 8.12b. The constructed path has $|R| + 1$ edges and $|R| + 2 \geq \lfloor n/2 \rfloor + 2$ points.

This concludes the proof. □

In Corollary 8.1 we noted that we can always find an orthogeodesic alternating path on the grid for any point set P such that each pair of points has a horizontal or vertical distance of at least two. Given an equitable point set P that does not satisfy this property, we can always find an equitable point set $P' \subseteq P$ such that P' has at least $|P|/16$ points and such that every pair of points in P' has a horizontal and a vertical distance of at least two, respectively. We can achieve this as follows. First, we sort the points from left to right and we remove every other point. Whenever we remove a point with color c we arbitrarily pick another point with color $c' \neq c$ and remove this point as well. The resulting point set remains equitable and contains at least $|P|/4$ points. We then repeat this process vertically such that the remaining point set has at least $|P|/16$ points. However, we can improve on this as the following theorem shows.

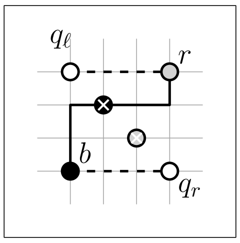
Theorem 8.7. *Let P be an equitable set of grid points. There is an $O(n \log^2 n)$-time algorithm that computes an equitable set $P' \subseteq P$ with $|P'| \geq |P|/3$ that admits a Hamiltonian orthogeodesic alternating path on the grid.*

Before we prove the theorem we prove the following auxiliary Lemma.

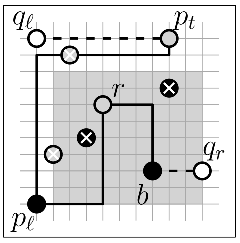
Lemma 8.3. *Let P be a balanced point set consisting of at least four and at most twelve points. Then P admits an orthogeodesic alternating path π satisfying the invariants* (H1) *and* (H2) *from the proof of Theorem 8.1 such that each point of π is charged with at most two points from P not on π.*

Proof. First, suppose that $4 \leq |P| \leq 6$. Then we can find a balanced subset P' of P consisting of at least four points. Suppose that the topmost point p_t is red and let p_b be the bottommost blue point. Then $y(p_t) - y(p_b) \geq 2$ and we can connect p_t to p_b by a vertical chain whose horizontal segment is one unit below p_t as illustrated in Figure 8.13a. If the bottommost point is red, then we can argue similarly. Finally, if both the topmost and the bottommost point is red, then let p_t be the topmost red point and let p_b be the bottommost blue point. Again we have $y(p_t) - y(p_b) \geq 2$ and we can connect p_t to p_b by a vertical chain whose horizontal segment is one unit below p_t. Clearly, the invariants *(H1)* and *(H2)* are maintained. Further, since P has at most six points and the constructed path π has two points, each of the two points can be charged with at most two points not on π as claimed.

Second, suppose that $8 \leq |P| \leq 12$. We show that we can find an alternating path with the desired properties consisting of at least four points. If $|P| \leq 12$, then such a path contains all but at most eight points. Thus, each point of the path can be charged with at most two



(a) $|P| = 4$



(b) Case 1.1.1

Figure 8.13: Illustrations for the proof of Lemma 8.3 according to the case distinction.

points not on the path. Without loss of generality we assume that $|P| = 8$. Otherwise we can always find a balanced subset of P consisting of eight points. We make a case distinction similar to the case distinction in the proof of Theorem 8.1. We let p_ℓ, p_r, p_t and p_b denote the leftmost, rightmost, topmost and bottommost points in P. Note, that some of these points may coincide.

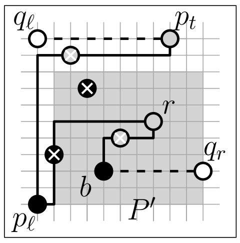
Case 1: The color of p_ℓ is blue. We distinguish three cases based on the colors of p_t and p_b, respectively.

Case 1.1: The color of p_t is red. Let p denote the point one unit below p_t if it exists. Let $P' := P \setminus \{p_\ell, p_t, p\}$. That is, P' is equitable and contains at least five points. Let the points in P' be denoted by the sequence $p_1, \ldots, p_k$, sorted from left to right and let σ denote the sequence of colors of $p_1, \ldots, p_k$ from left to right. We connect p_t to p_ℓ as illustrated in Figure 8.13b by a vertical chain whose horizontal segment is one unit below p_t. We distinguish two cases based on the sequence σ of colors.

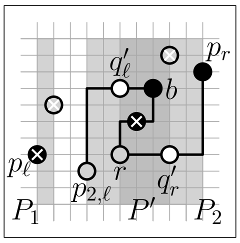
Case 1.1.1: The sequence σ contains a red point r followed by a blue point b. Then we connect r to b by an L-shaped orthogeodesic chain that is horizontally attached to r and vertically attached to b and we connect p_ℓ to r by an L-shaped orthogeodesic chain that is horizontally attached to p_ℓ and vertically attached to r as illustrated in Figure 8.13b.

Case 1.1.2: All blue points are at the beginning of the sequence and all red points are at the end of the sequence. First, suppose that the set P' contains three blue points and two red points. Then the point set $P' \setminus \{p_1\}$ contains a balanced set P'' of four points in which we can find a Hamiltonian alternating path consisting of two points r and b that satisfies the invariants *(H1)* and *(H2)*. Thus, we can connect p_ℓ to r by a horizontal chain whose vertical segment is on the vertical line through p_1 as illustrated in Figure 8.14a.

Second, suppose that P' contains three red points and two blue points. Let b be the bottommost blue point in $P' \setminus \{p_1\}$. Then there are either two red points above or two red points below b, respectively. Suppose that there are two red points above b. The other case can be handled similarly. Let r be the topmost red point. Then $y(r) - y(b) \geq 2$. Hence, we can connect r to b by a vertical chain whose horizontal segment is one unit below r as illustrated in



(a) Case 1.1.2



(b) Case 1.3.2a

Figure 8.14: Illustrations for the proof of Lemma 8.3 according to the case distinction.

Figure 8.14a. Finally, we connect p_ℓ to r by a horizontal chain whose vertical segment is on the vertical line through p_1.

Case 1.2: The color of p_b is red. This case is symmetric to Case 1.1.

Case 1.3: The color of p_t is red and the color of p_b is red. We distinguish two cases based on the color of p_r.

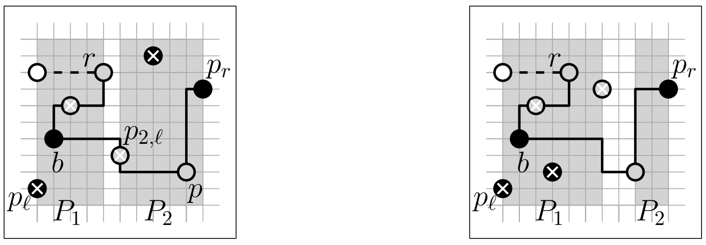
Case 1.3.1: The color of p_r is red. This case is analogous to Case 1.1, except that we attach the newly created edges at the blue end of the path instead of at the red end of the path.

Case 1.3.2: The color of p_r is blue. By symmetry we can find a horizontal partition of P into two balanced point sets P_1 and P_2, respectively such that the leftmost point $p_{2,\ell}$ of P_2 is red. We distinguish three cases based on the size of $|P_2|$.

Case 1.3.2a: $|P_2| = 6$. Let $P' := P_2 \setminus \{p_{2,\ell}, p_r\}$ as illustrated in Figure 8.14b. Then P' is balanced and contains at least four vertices and we can find an orthogeodesic alternating path consisting of two vertices r and b such that both the horizontal segment $\overline{q'_\ell b}$ and the horizontal segment $\overline{rq'_r}$ do not intersect the edge between r and b, where q'_ℓ denotes the point on the left side of $\mathcal{B}(P')$ that is horizontally aligned with b and q'_r denotes the point on the right side of $\mathcal{B}(P')$ that is horizontally aligned with r. That is, we can connect $p_{2,\ell}$ to b by an L-shaped orthogeodesic chain that is vertically attached to $p_{2,\ell}$ and horizontally attached to b and we can connect r to p_r by an L-shaped orthogeodesic chain that is horizontally attached to r and vertically attached to p_r as illustrated in Figure 8.14b. Then we have constructed a path consisting of four vertices.

Case 1.3.2b: $|P_2| = 4$. Then $|P_1| = 4$ and we can find a path consisting of a red point r and a blue point b with the desired properties in P_1. Let p be the rightmost red point in P_2. Then we can connect b to p by a horizontal chain whose vertical segment is on the vertical line of the leftmost red point $p_{2,\ell}$ in P_2 as illustrated in Figure 8.15a. Note that, $p_{2,\ell} \neq p$ since P_2 contains 2 red points of which p is the rightmost.

Case 1.3.2c: $|P_2| = 2$. Then $|P_1| = 6$ and the leftmost five points in P_1 must contain a balanced point set P'_1 of four vertices. Then we can find an



(a) Case 1.3.2b (b) Case 1.3.2c

Figure 8.15: Illustrations for the proof of Lemma 8.3 according to the case distinction.

orthogeodesic chain in P_1' connecting a red point r and a blue point b in P_1'. Then we connect $p_{2,\ell}$ to p_r by an L-shaped chain that is horizontally attached to $p_{2,\ell}$ and vertically attached to p_r and we connect b to $p_{2,\ell}$ by a horizontal chain whose vertical segment is on the vertical line through the rightmost point of P_1 as illustrated in Figure 8.15b.

Case 2: The color of p_ℓ is red. We distinguish two cases based on the color of p_r.

Case 2.1: The color of p_r is blue. Note that the point set P has the same properties as the point set P' in Case 1.3.2a. Therefore, we can handle this case in exactly the same way illustrated in Figure 8.14b, except that need to charge two points less to the computed path.

Case 2.2: The color of p_r is red. This case is analogous to Case 1.3.2, except that we exchange the roles of P_1 and P_2.

In all cases we constructed a path with four vertices satisfying the invariants *(H1)* and *(H2)*. Since P contains at most twelve vertices, at most eight vertices are not contained in the path. Hence, each point of the path can be charged with at most two of these points. □

Now we turn to the proof of Theorem 8.7.

Proof of Theorem 8.7. We slightly modify the algorithm described in Section 8.2 by forcing it to use only grid lines and by allowing it to remove points during the execution if they would obstruct an edge that the algorithm as suggested for Corollary 8.1 would draw next. To keep the point set balanced, we always remove pairs of points with different colors. We show that each point on the path computed by the algorithm can be charged with at most two removed points. That is, the length of the computed path is at least $|P|/3$.

Our proof is based on a case distinction similar to the proof of Theorem 8.1. First, note that we can find an orthogeodesic alternating path on the grid containing two vertices for every equitable point set with at most three points. We simply connect an arbitrary red point to an arbitrary blue point by an L-shaped edge. Thus, we only consider point sets with at least four points.

First, we consider only the balanced case. That is, for every balanced point set consisting of $n = 2k$ with $k \geq 1$ such that no pair of points is horizontally or vertically aligned, we prove that we can compute a Hamiltonian orthogeodesic alternating path on the grid containing at least $|P|/3$ points such that the invariants *(H1)* and *(H2)* according to the proof of

Theorem 8.1 are maintained and such that each point on the path is charged at most two points that are not on the path. The proof is by induction on n. We let the base cases be all balanced point sets with at least four and at most twelve points. By Lemma 8.3 we can always find an orthogeodesic path with the desired properties on such a point set.

Next, suppose that the induction hypothesis holds for all $2k < n$ such that $k \geq 2$ and $n \geq 14$. We make a case distinction according to the proof of Theorem 8.1 and as illustrated in Figure 8.4. Additionally, we use the definitions and terminology according to the proof of Theorem 8.1. That is, by p_ℓ, p_r, p_b and p_t we denote the leftmost, rightmost, bottommost and topmost points in P, respectively.

Case 1: The color of p_ℓ is blue. We distinguish three sub-cases.

Case 1.1: The color of p_t is red. Let $Q \subseteq P$ be the points distinct from p_ℓ and p_t on the horizontal line one unit below p_t and on the vertical line one unit to the right of p_ℓ, respectively, that is $|Q| \leq 2$ and $p_\ell, p_t \notin Q$.

Let Q' be an arbitrary set of two points from $P \setminus (Q \cup \{p_\ell, p_t\})$ such that $P' := P \setminus (Q \cup Q')$ is balanced. Then P' contains at least eight points and we can apply the induction hypothesis to P'. The at most four points in $Q \cup Q'$ will be charged to the two new vertices on the path.

We connect p_ℓ to the path computed in P' as in Case 1.1 in the proof of Theorem 8.1 and as illustrated in Figure 8.1b.

Case 1.2: The color of p_t is blue and the color of p_b is red. This case is obtained from Case 1.1 by a reflection.

Case 1.3: The color of p_t is blue and the color of p_b is blue. We consider two sub-cases depending on the color of p_r.

Case 1.3.1: The color of p_r is red. This case is similar to Case 1.1. Instead of attaching two new points to the left side of the path, we attach two more edges at the right side of the path. Clearly, we must remove at most four points for two newly created edges.

Case 1.3.2: The color of p_r is blue. Let P_1 and P_2 be a partition of P according to Lemma 8.1. By symmetry we can choose P_2 such that the leftmost point $p_{2,\ell}$ in P_2 is red. Suppose that the vertical line one unit to the left of $p_{2,\ell}$ is occupied by a point p. Clearly, $p \in P_1$.

First, suppose that P_1 contains at most four vertices. Then P_2 contains at least ten vertices and we can handle P_2 according to Case 2.1 without charging the first and the last point of the resulting path with any removed points since we can use L-shaped edges incident to these points, which do not cross any other points. Note that the point set $P_2' := P_2 \setminus \{p_{2,\ell}, p_r\}$ still contains at least eight points such that we can apply the induction hypothesis on P_2' after handling P_2 according to Case 2.2. That is, we can remove all points in P_1 and charge the removal to p_ℓ.

Otherwise P_1 contains at least six points, that is, we can remove p and an arbitrary point p' whose color is different from $c(p)$ and apply the induction hypothesis to the set $P_1' := P_1 \setminus \{p, p'\}$ since $|P_1'| \geq 4$. If P_2 has only two vertices, we directly connect $p_{2,\ell}$ to p_r by an L-shaped edge and charge the removal of p and p' to $p_{2,\ell}$. If P_2 has four vertices, then we can also connect $p_{2,\ell}$ to p_r by an L-shaped edge and charge the removal of p, p' and the remaining two points in $P_2 \setminus \{p_{2,\ell}, p_r\}$ to $p_{2,\ell}$ and p_r, respectively. Finally, if P_2 contains at least six vertices, we can handle P_2 according to Case 2.1 without charging $p_{2,\ell}$ and p_r since we can apply the induction hypothesis to the point set $P' := P_2 \setminus \{p_{2,\ell}, p_r\}$ containing at least four vertices.

The sub-paths constructed for P_1 and P_2, if any, are connected according to Case 1.3.2 of the proof of Theorem 8.1 and as illustrated in Figure 8.3b.

Case 2: The color of p_ℓ is red. We consider two sub-cases.

Case 2.1: The color p_r is blue. Note that P has at least 14 points, that is we can apply the induction hypothesis to $P \setminus \{p_\ell, p_r\}$. Then we connect p_ℓ to b' and r' to p_r using L-shaped edges. Clearly, this satisfies invariant *(H1)* since p_ℓ and p_r at on the left and right side of the bounding box of P, respectively.

Case 2.2: The color of p_r is red. This case is similar to Case 1.3.2, except that we exchange the roles of P_1 and P_2.

The unbalanced case can be handled similar to the proof of Theorem 8.1. Suppose that P is an unbalanced equitable point set consisting of at least 5 red and blue points. We may assume without loss of generality that $|B| = |R| + 1$. First, consider the case that one of the points p on the boundary of $\mathcal{B}(P)$ is blue. Assume without loss of generality that p is on the left side. Then we can compute a path as described earlier for the balanced set of points $P' := P \setminus \{p\}$ and connect p to this path by an L-shaped orthogeodesic chain that is vertically attached to p and horizontally attached to the red end of the path computed for P'.

Second, consider the case that all points on the boundary are red. Then we add a new red point r to the left of P and consider the point set $P' := P \cup \{r\}$. Then P' has at least six points such that the leftmost two points are red. We split P' into two point sets P_1 and P_2 according to Lemma 8.1 such that the rightmost point $p_{1,r}$ in P_1 is blue. Clearly, P_1 contains at least four points, since the leftmost two points are red. First, suppose that P_1 contains exactly four points $p_1, \ldots, p_4$ sorted from left to right. Then the color of $r = p_1$ and p_2 is red and the color of p_3 and p_4 is blue, respectively. If P_2 contains only two vertices, then P contains only six points. Then we can pick any balanced subset of points from P and compute a an orthogeodesic alternating path with two points and we are done. If P_2 contains at least four points, we can compute a path in P_2 the by induction hypothesis starting with a red point r_2 and ending in a blue point b_2. Then we connect p_2 to p_3 by an L-shaped chain that is vertically attached to p_2 and horizontally attached to p_3 and we connect p_3 to r_1 by an L-shaped chain that is vertically attached to p_3 and horizontally attached to r_2. The removed points p_1 and p_4 are charged to p_3. Second, suppose that P_1 contains at least six points. Then we can handle P_1 according to Case 2.1 without charging r with the removal of any point. Finally, since r is only used as and endpoint of the path, if it is used at all, and since it is not charged with the removal of any points, we can safely remove it again.

Note that we can decide which points to remove before recursing on the point sets from which we removed the points. Therefore, the rest of the algorithm can be implemented and analyzed as in the proof of Theorem 8.1. Thus, the algorithm can be implemented to run in $\mathcal{O}(n \log^2 n)$ time. □

8.5 Concluding Remarks

In this chapter, we studied the existence of Hamiltonian orthogeodesic alternating paths. While point sets that do not admit a Hamiltonian orthogeodesic alternating path can trivially be constructed—as, for instance, any set of horizontally aligned point set whose points are not alternatingly red and blue—we proved that such a path can always be computed in $\mathcal{O}(n \log^2 n)$ time on every point set whose points are neither horizontally or vertically aligned. The constructed path needs two bends on some of the edges, which we proved to be worst-case optimal by presenting a family of graphs that does not admit a Hamiltonian orthogeodesic alternating path with at most one bend per edge. However, if we require both points and bends to be placed on the grid, then we proved that this problem is NP-complete. Using similar techniques we then showed that it is also NP-hard to decide whether a given set of points on the grid admits an alternating perfect matching if points are allowed to be be horizontally or vertically aligned. However, Kano [Kan09] had previously shown that such a perfect matching can always be constructed on the grid if points are not allowed to be horizontally or vertically aligned. Motivated by the hardness of deciding whether a given point set admits a Hamiltonian orthogeodesic alternating path on the grid we studied the problem of finding an orthogeodesic alternating path on the grid with maximum length and we presented a factor-3 approximation algorithm whose running time is $\mathcal{O}(n \log^2 n)$. In contrast to this, we showed that there are point sets for which there does not exist an orthogeodesic alternating path on the grid containing much more than half the number of points.

Open problems The algorithm we presented for computing a Hamiltonian orthogeodesic alternating path off the grid needs two bends on some of the edges. While it is not always possible to construct a path with at most one bend per edge, it would be interesting characterize the point sets admitting such a path and to devise an efficient algorithm for computing such a path, if it exists. Further, it is an interesting open problem to study the gap between the factor-3 algorithm approximation of the longest orthogeodesic alternating path and the presented upper bound for the worst-case ratio of an approximation algorithm.

Finally, there are several variations of the problem that might be interesting for future work. First, we can consider two-colored orthogeodesic point set embedding problems for various other classes of graphs, such as cycles, trees and planar or outerplanar graphs. Second, we can consider similar problems for more than two colors. *Straight-line* alternating paths on multi-colored point sets have been studied by Merino et al. [MSU06] but orthogeodesic alternating paths do not seem to have been considered on multi-colored point sets as of yet. And third, it would also be interesting to study a variant of the problem in which both the points in the plane and the vertices of the graph are colored and we ask for a point-set embedding of the graph such that a colored vertex may only be mapped to a point with the same color. We may think of the colors as encoding different functionalities of the vertices in the network. This problem is already capable of modeling complex constraints for the placement of the vertices.

Chapter 9

Generalizing Geometric Graphs

Network visualization is essential for understanding the data obtained from huge real-world networks such as flight-networks, the Internet or social networks. Although we can compute layouts for these networks reasonably fast, even the most recent display media are not capable of displaying these layouts in an adequate way. Moreover, the human viewer may be overwhelmed by the displayed level of detail. The increasing amount of data therefore requires techniques aiming at a sensible reduction of the visual complexity of huge layouts.

We consider the problem of computing a generalization of a given layout reducing the complexity of the drawing to an amount that can be displayed without clutter and handled by a human viewer. That is, we consider the geometric network construction problem of constructing a small network that is to resemble a given large network both visually and structurally. We take a first step at formulating graph generalization within a mathematical model and we consider the resulting problems from an algorithmic point of view. Although these problems are NP-hard in general, we provide efficient approximation algorithms as well as efficient and effective heuristics. At the end of the chapter we showcase some sample generalizations. This chapter is based on joint work with Edith Brunel, Andreas Gemsa, Ignaz Rutter and Dorothea Wagner [BGK$^+$12].

9.1 Introduction

As a natural consequence of the increasing amount of available data we are frequently facing large and even huge networks such as road and flight networks, the Internet and social networks with millions of vertices. Visualization of these networks is a key to assessing the inherent graph-based information via human inspection. There are several methods for computing layouts of huge graphs with millions of vertices within a few minutes [HK02b, KCH03, HJ05].

But, how do we display such layouts? Modern HD displays feature only roughly 2 Mio pixels and a standard A4 page allows only roughly 8.7 Mio dots at a resolution of 300 pixels per inch. Although these numbers do sound adequate for large-scale graph visualization at first glance, both media are not at all suited for displaying huge graphs with millions of vertices. Even if we require only a minimal distance of 10 pixels or dots between the vertices of the graph, which yields a distance between vertices of roughly 3 millimeters on the screen and less than 1 millimeter on paper, then we can display only several thousand vertices, and not too many edges. If we additionally seek to display graph structure and keep visual clutter low, the number of vertices we can display degrades even further and may go down as far as less than a hundred for dense graphs.

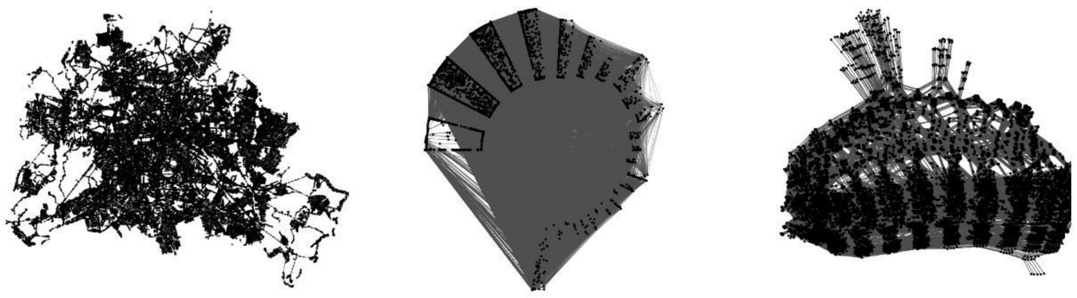
Even worse, the human perception is not capable of extracting detailed information from huge layouts with millions of vertices. Since, by a simple counting argument, there are incompressible adjacency matrices [LV08], a graph with only 1 Mio vertices may encode incompressible information of up to 125 Gigabytes. This exceeds by a factor of 3.6 the average daily information consumption of an average American estimated at 34 (highly compressible) Gigabytes of information in the current report on American Consumers [BS09].

Related Work Known approaches to coping with the huge amount of data by allowing for some kind of abstraction can be categorized into *structural* and *geometric* methods. While structural methods create a new layout for the data, typically using a clustering of the graph, geometric methods are applied to a given layout maintaining the user's mental map [MELS95].

Graph-theoretic clustering methods, which can be used to cluster the graph for visualization are discussed in [Gae05]. Eades and Feng [EF97] describe a multilevel visualization method for clustered graphs with the aim of visualizing network-based data that has been clustered hierarchically at different levels of abstraction induced by the hierarchy of the clustering. A force-directed layout algorithm based on a hierarchical decomposition of the graph is given by Quigley and Eades [QE01]. This method allows for visualizing the graph at different levels of abstraction by computing a layout based on a hierarchical grouping of the vertices of the graph. Harel and Koren [HK02a] present a multi-scale algorithm with the purpose of producing nice drawings on large and small scale, respectively. Different levels of abstraction of the graph are obtained by iteratively coarsening the graph. Multi-scale drawing methods are combined with fisheye views by Gansner et al. [GKN05]. Their approach is to compute a layout of a graph whose level of detail deteriorates with increasing distance to the focal node of the layout, that is, they provide a topological version of classical fisheye visualization techniques. Abello el al. [AKF01] discuss graph sketches for very large graphs based on mapping clusters of the graph to certain regions of the screen. Their notion of a sketch is based on a hierchcical clustering of the graph and is mainly focused at exploring the graph via a detail-on-demand strategy without providing a good approximation of the graph's structure and geometry. Rafiei and Curial [RC05] study the generalization of graphs by sampling.

Classical fisheye visualizations [Fur86, SB92], on the other hand, can be directly applied to a given layout and apply a distortion to a given layout to emphasize the structure of the drawing in a certain area of interest. The resolution of the drawing deteriorates towards the boundary of the drawing and parts of the drawing in this area are usually densely cluttered. Abello et al. [AKY05] study the visualization of large graphs with compound-fisheye views and treemaps, employing hierarchical clustering and a treemap representation of this clustering. Further, edge bundling techniques [TE10, HvW09] aim at reducing the complexity of layouts by bundling similar edges.

Generalization has received considerable attention in cartography [MRS07]. Apart from this, Mackaness and Bear [MB93] highlight the potential of graph theory for map generalization. Saalfeld states the map generalization problem as a straight-line graph drawing problem [Saa95] and formulates a number of challenges resulting from this perspective. Among others, he asks for a rigorous mathematical model for graph-based generalizations and provable guarantees. We are not aware of any work aiming at assessing this problem to its full extent.



(a) Vertex-Clutter (b) Edge-Clutter (c) Vertex-Edge-Clutter

Figure 9.1: Illustration of different types of clutter.

Contribution We take a first step towards establishing a mathematical model for the problem of generalizing geometric graphs. A key to assessing the problem of computing a suitable generalization is to find an adequate measure of the quality or appropriateness of a generalization. It is essential to understand the geometric and combinatorial features resulting in the visual complexity of geometric graphs and how they affect human perception. The *geometric features* of the drawing include, among others, the distributions of points and edges as well as the distribution of crossings, the shapes of the faces of the arrangement, especially the outer face, as well as symmetries and peculiarities. The *combinatorial features* include connectivity, structure and length of shortest paths as well as, for instance, planarity. Although we are far from fully understanding the impact of the these features on the human perception we try to incorporate a carefully selected set of these features into our model of a generalization. The generalization should maintain the spirit of the drawing of the graph and preserve the prominent features while reducing the amount of detailed information to an amount that can be displayed without clutter and handled by a human viewer. Our model is based on the fact that vertices have a fixed size and edges have a fixed width on the screen. *Visual clutter* refers to an agglomeration of overlapping visual features in a limited area that renders these features indistinguishable. Our goal is to either avoid or reduce visual clutter. We identify three types of clutter.

Vertex-Clutter occurs when two or more vertices are too close to each other. It may render the drawing unusable due to hidden edge information as illustrated in Figure 9.1a.

Edge-Clutter occurs when too many edges cross a limited area. Even if vertices are far enough apart, edge clutter may lead to indistinguishable edge information as illustrated in Figure 9.1b.

Vertex-Edge-Clutter occurs when a vertex is too close to an edge. In this case, we are unable to tell, whether the vertex is incident to the edge or not as illustrated in Figure 9.1c.

We devise a framework that allows for assessing all types of clutter in an incremental way by modeling the elimination or reduction of each type of clutter as an optimization problem, which we analyze in terms of complexity. We show that these problems are NP-hard in general and we provide approximation algorithms as well as effective and efficient heuristics that can be applied to huge graphs within reasonable time.

Preliminaries In this chapter, we model a *geometric graph* as a pair $G = (P, E)$ such that $P \subseteq \mathbb{R}^2$ is a finite set of n points in the plane and E is a set of m straight-line segments with endpoints in P. If not otherwise stated, graph refers to a geometric graph throughout this chapter. For $p \in P$ and a non-negative number $r \in \mathbb{R}_0^+$, we denote by $B(p, r)$ the disk with center p and radius r. We model the finite resolution of a screen by assuming that each point p occupies the locus of points whose distance to p is bounded by $s \in \mathbb{R}_0^+$ and, similarly, each edge e occupies the locus of points whose distance to e is bounded by $w \in \mathbb{R}_0^+$.

A *generalization* of G is a pair (H, φ) where $H = (Q, F)$ is a geometric graph with $Q \subseteq P$ such that $\varphi\colon P \to Q$ maps vertices of G to vertices of H and F is a subset of edges resulting from a contraction of G according to φ. Since the subgraph induced by $\varphi^{-1}(q)$ is contracted into a single vertex, we call this subgraph the *cluster* of q, denoted by C_q. Given $Q \subseteq P$, we denote by $\nu\colon P \to Q$ the *Voronoi mapping*, which maps $p \in P$ to its closest neighbor in Q with respect to the Euclidean metric. We call the corresponding clusters *Voronoi clusters*. We especially focus on this mapping since it minimizes the sum of the distances $\sum_{p \in P} d(p, \varphi(p))$ between the original vertices and the points to which they are mapped in the generalization. Hence, this mapping seems to be a natural choice. Throughout the chapter distance refers to the Euclidean metric.

Organization of the Chapter In Section 9.2, we consider the problem of eliminating vertex-clutter. We discuss our model for the generalization of the vertex set and show NP-hardness of the corresponding optimization problem. We further show that the size of the generalized point set can be approximated efficiently and we devise an efficient heuristic for further optimization. In Section 9.3, we study the reduction of edge-clutter. We show that it is in general NP-hard to find a sparse or short subset of the edges maintaining monotone tendencies. When the original graph is complete, however, or if we are not restricted to use edges of the original graph, we can efficiently compute a sparse graph approximately representing monotone tendencies of the edges. In Section 9.4, we model the problem of reducing vertex-edge clutter and we show how to compute a drawing that allows for unambiguously deciding whether an edge is incident to a vertex or not, thus effectively eliminating vertex-edge clutter. We showcase and discuss some sample generalizations in Section 9.5.

9.2 Generalizing the Vertex Set without Vertex-Clutter

In this section we consider the problem of computing a generalization (H, φ) without vertex clutter for a geometric graph $G = (V, E)$, where $H = (Q, F)$. We focus on the case that φ is the Voronoi-mapping assigning each vertex in P to its nearest neighbor in Q. In order to avoid vertex-clutter, we require a minimal distance $r \in \mathbb{R}_0^+$ between the vertices of a generalized geometric graph. Let $\varrho\colon P \to \mathbb{R}_0^+$ be a function that maps a positive real number $\varrho(p) \geq r$ to every point $p \in P$. For each vertex $p \in Q$ in the generalized graph we require that the disk $B(p, \varrho(p))$ does not contain any other point from Q. We call a point set Q with this property a *ϱ-set of P*. This prerequisite, however, must be balanced with additional quality measures such as the size of the ϱ-set, the clustering induced by φ and the distribution of the points in Q in order to avoid trivial solutions such as a single vertex. Clearly, it is desirable to maximize the size of a ϱ-set in order to retain as many vertices of the original graph as possible. That is, even in the presence of other optimization goals we may assume that the vertex set Q of the generalization constitutes an inclusion-maximal ϱ-set of the original point set P.

Choosing $\varrho \equiv r$ uniformly for all points $p \in P$ may have a severe effect on the distribution of the points when maximizing the size of a ϱ-set since the distances to the nearest neighbors in an inclusion-maximal ϱ-set tend to be uniformly distributed regardless of the original distribution. However, it may be more appropriate to approximate the distribution of the original point set. In order to approximate this distribution by an inclusion-maximal ϱ-set we can choose ϱ as follows. Let p_0 be the point that maximizes the number of points in $B(p, r) \cap P$ over all $p \in P$ and let $k = |B(p_0, r) \cap P| - 1$ denote the number of points in this disk that are different from p_0. For each $p \in P$ let $d_k(p) \geq r$ denote p's distance to its k-nearest neighbor in P. By choosing $\varrho(p) = d_k(p) \geq r$ any inclusion-maximal point set will have approximately the same distribution as the original point set since for each point in the generalized point set we discarded the same amount of points from the original graph.

Since, in general, it is not clear which behavior is more appropriate, we introduce a parameter $\alpha \in [0, 1]$ and let the user decide by setting $\varrho(p) := \max\{r, \alpha d_k(p)\}$. That is, the user can choose between retaining as many points in areas with low clutter as possible ($\alpha = 0$) and approximating the distribution of the point set ($\alpha = 1$) as well as interpolations between the two extremes.

We consider two measures to assess the quality of a ϱ-set Q. While the size of Q is a measure of the amount of data that is retained, the quality of the clustering induced by φ is a measure for the amount of data that is lost due to the contraction of the vertices. There are several established ways of assessing the quality of clusterings, such as coverage, performance, conductance [Gae05], and modularity [BDG+08]. Since the information contained in the inter-cluster edges is retained in the generalization, we concentrate on assessing the quality of the clusters based on the intra-cluster edges. We consider a measure similar to coverage, which we adapt to our purpose as follows. For each cluster C_q let n_q denote the number of vertices and m_q denote the number of edges in C_q, respectively. We define the *local coverage* of a cluster C_q by $\operatorname{lcov}(C_q) = 2m_q/(n_q(n_q - 1))$, that is, as the amount of intra-cluster coherence that is explained by the intra-cluster edges. The local coverage of the generalization is defined as $\operatorname{lcov}(H, \varphi) = \min_{q \in Q} \operatorname{lcov}(\varphi^{-1}(q))$.

In order to reduce vertex-clutter, we consider the following multi-objective optimization problem. Given a geometric graph $G = (P, E)$, a non-negative radius $r \in \mathbb{R}_0^+$ and $\alpha \in [0, 1]$ the Local Coverage Cluster Packing (LCCP) problem is to compute a ϱ-set $Q \subseteq P$ and a mapping $\varphi \colon P \to Q$ that maximizes both $|Q|$ and $\operatorname{lcov}(H, \varphi)$.

Problem Local Coverage Cluster Packing (LCCP)

Instance: Geometric graph $G = (P, E)$, $r \in \mathbb{R}_0^+$, $\alpha \in [0, 1]$

Solution: ϱ-set $Q \subseteq P$, mapping $\varphi \colon P \to Q$

Goal: maximize $\operatorname{lcov}(H, \varphi)$, maximize $|Q|$

First we show that several single-criteria optimization variants of this multi-criteria optimization problem are NP-hard. Then we show how to approximate the size of a ϱ-set efficiently and we devise an efficient heuristic for balancing the size of a ϱ-set with the quality of the induced local coverage.

9.2.1 Complexity

The problem of computing a ϱ-set of maximum size for $\alpha = 0$ can be reduced to the problem of computing a maximum independent set in the intersection graph of the disks with radius $r/2$ centered at the points in P. Clark et al. [CCJ90] prove that this problem is NP-hard in unit-disk graphs, even if the disk representation of the graph is given.

Corollary 9.1. *Maximizing the size of a ϱ-set is NP-hard for $\alpha = 0$.*

Next, we show that it is also NP-hard to maximize the local coverage in the induced clusters of a ϱ-set as well as the total size of the generalization obtained by choosing a ϱ-set if the clustering is obtained by the Voronoi mapping induced by the points in Q.

Theorem 9.1. *Maximizing* $\mathrm{lcov}(H, \nu)$ *of a generalization* (H, ν) *is NP-hard for* $\alpha = 0$.

Proof. We prove the theorem by reduction from the NP-hard problem Planar Monotone 3-sat [dBK10]. Let $\mathcal{U} = \{x_1, \ldots, x_n\}$ be a set of Boolean variables and let $\mathcal{C} = C_1 \wedge C_2 \cdots C_m$ be 3-sat formula. Then $\mathcal{C}$ is called *monotone* if all clauses consist only of positive or only of negative literals. Let $\mathcal{G} = (\mathcal{U} \cup \mathcal{C}, \mathcal{E})$ be the bipartite graph, on the clauses and variables, where $\mathcal{E}$ contains the edge (x_i, C_j) if and only if the literal x_i or its negation is contained in C_j. A *monotone rectilinear representation* of a monotone 3-sat formula is a rectilinear drawing of $\mathcal{G}$ such that the following conditions are met, as illustrated in Figure 9.2.

(i) The variables and clauses are drawn as axis-aligned boxes such that all variable boxes are on a horizontal line.

(ii) The edges are drawn as vertical line segments connecting the corresponding boxes.

(iii) The drawing does not contain any crossings.

An instance of Planar Monotone 3-sat consists of a *monotone rectilinear representation* of a planar monotone 3-sat instance and we wish to decide, whether the corresponding 3-sat instance is solvable.

A ϱ-set with local coverage 1 is called a *perfect* ϱ-set. A ϱ-set is perfect if and only if the graphs induced by the vertices in each of the Voronoi faces defined by Q are cliques. Given a monotone rectilinear representation of a planar monotone 3-sat formula we will construct a corresponding instance $I = (G = (P, E), \varrho)$ of problem LCCP such that I contains a perfect ϱ-set if and only if the 3-sat formula is satisfiable. For reasons of simplicity our

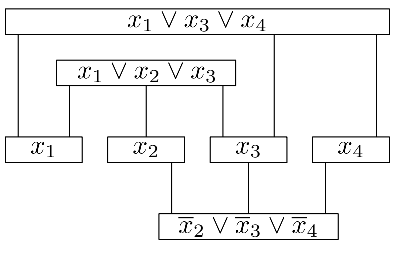


Figure 9.2: Monotone rectilinear representation of the 3-sat formula $(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_3 \vee x_4) \wedge (\overline{x}_2 \vee \overline{x}_3 \vee \overline{x}_4)$.

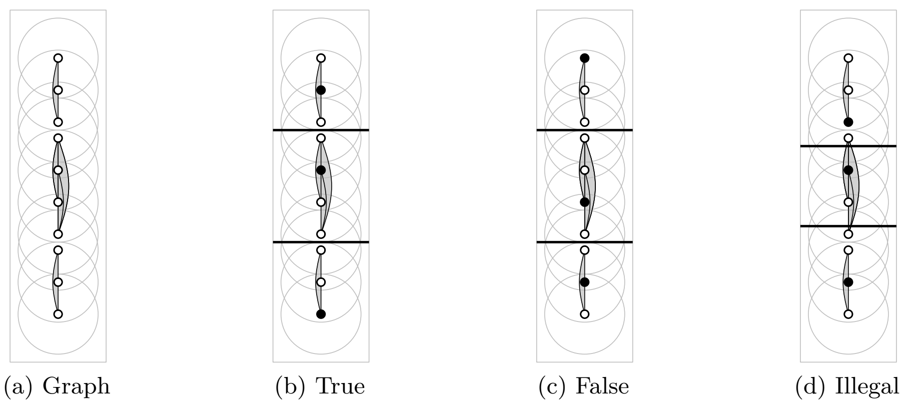

(a) Graph (b) True (c) False (d) Illegal

Figure 9.3: Variable gadget

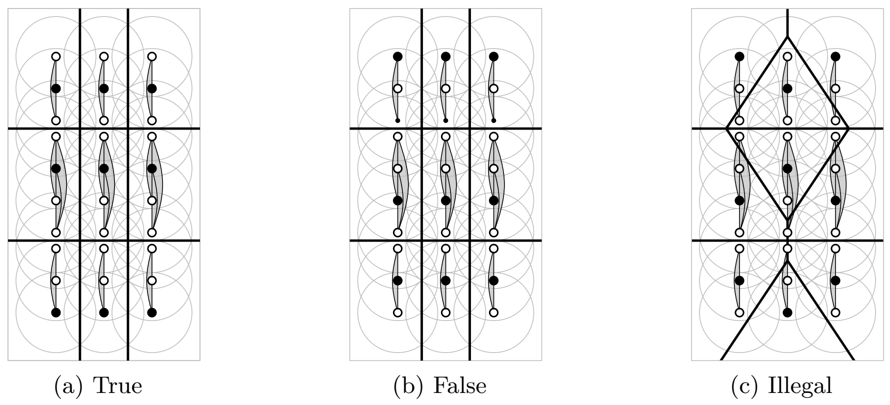

(a) True (b) False (c) Illegal

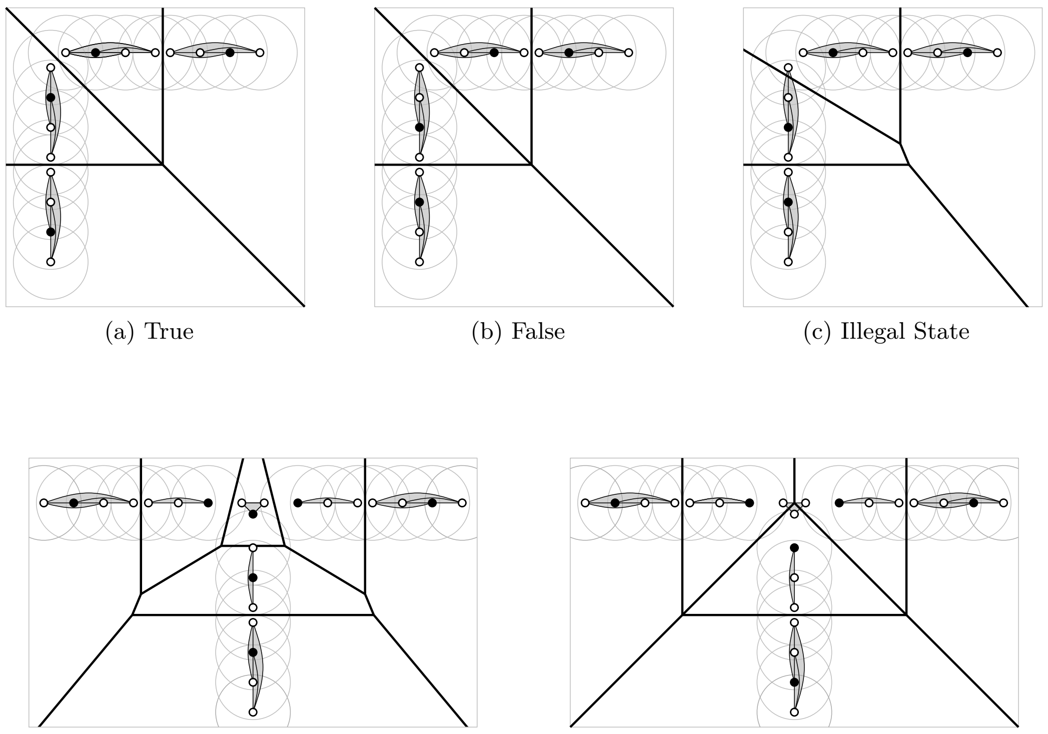
Figure 9.4: Literal gadget

construction is based on a disconnected graph with collinear points, but the construction can be modified in a straightforward way to obtain similar results for connected graphs with vertices in general position. We choose $\varrho \equiv 1.25$ and we construct G from a set of *variable/literal gadgets*, *transmitter/bend gadgets* and *clause gadgets*, which we will describe subsequently. We will use the following trivial observation.

Observation 9.1. Every perfect ϱ-set of G contains at least one vertex of each clique of G.

Variable/Literal Gadget. We distinguish between *basic* and *extended* variable and literal gadgets, respectively. The basic variable gadgets incorporate the functionality needed to correctly represent the variables. These gadgets can be extended to transmit their state along the transmitters. Each basic variable gadget consists of three vertically aligned cliques of size three and four, respectively, as illustrated in Figure 9.3. Each clique consists of vertically aligned points at distance 1, and the cliques are separated by a vertical gap of 0.5.

Due to Observation 9.1 a perfect ϱ-set of G must contain at least one vertex in each of the cliques. Note that we cannot choose two vertically subsequent vertices in any of the cliques, since they do not constitute a ϱ-set. Due to the chosen vertical distribution of the vertices, the vertices of any ϱ-set closest to the gaps must be chosen symmetrically to the bisector of the two cliques. Otherwise, the bisector of these vertices will intersect the edges of one of the


(a) True

(b) False

(c) Illegal State

(d) satisfied

(e) dissatisfied

Figure 9.5: (9.5a)–(9.5c) Bend gadget, (9.5d)–(9.5e) Clause gadget.

cliques as illustrated in Figure 9.3d. Furthermore, we cannot choose the two vertices closest to the gap since they do not constitute a valid ϱ-set. Hence, there are only two valid ϱ-sets of the basic variable gadget, corresponding to the true and false state of the corresponding variable, as illustrated in Figure 9.3b and 9.3c, respectively. The variable gadgets can be extended in order to connect them to the transmitter gadgets. In order to extend the variable gadgets, we substitute the cliques of size 3 at the corresponding end by a clique of size 4.

The literal gadgets are composed of basic and extended variable gadgets that are horizontally aligned. The horizontal gap between the gadgets is variable and can be chosen to be 1 or 1.5, as illustrated in Figure 9.4.

Transmitter/Bend Gadget. The transmitter gadgets consist of two vertically aligned cliques of size 4 that are separated by a gap of 0.5 similar to the variable gadgets. When stacked upon the extended variable gadgets with a vertical gap of 0.5 the transmitter gadgets can be in one of two valid states corresponding to the assignment of the variable. The bend gadgets consist of one vertical and one horizontal transmitter gadget as illustrated in Figure 9.5. Figure 9.5c illustrates that the state cannot change at the transition between the horizontal and the vertical transmitter segment. Such a change would result in locate coverage strictly smaller than 1.

Clause Gadget. Finally, the clause gadget is constructed as illustrated in Figure 9.5. It consists of three small gadgets that are arranged in a T-shaped fashion and which are constructed exactly as the topmost two cliques of the basic variable gadget. The functionality of the clause is realized by a small triangle arranged in the middle of the T-shaped figure. If all literals corresponding to the clause are false, then each of the triangle's vertices is contained in the ϱ-ball of one of the vertices contained in the corresponding ϱ-set. Hence, none of the vertices of the triangle may be contained in the ϱ-set and, thus, the vertices of the triangle are mapped to a neighboring point resulting in local coverage strictly less than 1. This is illustrated in Figure 9.5e. If, on the other hand, at least one of the variables is in a true state, then one of the triangle's vertices can be included in the ϱ-set. The vertices of the triangle are chosen in such a way that the bisector between any of these vertices and the closest vertex corresponding to a true assignment does not intersect any of the cliques of the clause. As illustrated in Figure 9.5d, this leads to a Voronoi diagram that does not intersect the edges of G, resulting in a perfect ϱ-set.

Clearly, a satisfying assignment of the 3-SAT formula can be transformed into a perfect ϱ-set of G. Conversely, assume that we are given a perfect ϱ-set of G. As argued, the variable gadgets can be in one of two states in this case as illustrated in Figure 9.3. This state is likewise represented in the adjacent transmitters and will thus be transmitted without error to the clauses. Since the ϱ-set is perfect, one of the central triangle's vertices of each clause gadget must be in the set. Hence, at least one of the adjacent transmitters must be in a true state, corresponding to the assignment of one of the literals. Hence the states of the variables as in Figure 9.3 correspond to a satisfying assignment of the 3-SAT formula. A sample reduction is illustrated in Figure 9.6.

Since the reduction is based on deciding whether the given graph contains a perfect ϱ-set whose size is equal to the number of cliques in G it yields that both the maximization of local coverage as well as the maximization of the size of the ϱ-set with given minimum local coverage are NP-hard.

□

9.2.2 Approximating the Maximum Size of a Generalization

Although it is unlikely that we can efficiently compute a ϱ-set with maximum size, we show that we can approximate the size of a maximum ϱ-set.

Theorem 9.2. *Let G be a geometric graph and let $r \in \mathbb{R}_0^+$ and $\alpha \in [0,1]$ be given. In $\mathcal{O}(kn + n\log^5 n(\log\log n)^2)$ time we can compute a generalization $\mathcal{H}$ of G that approximates the maximum number of vertices of a generalization by a factor of $(7k+2)/3$, where $k = \max_{p\in P} |B(p,\varrho(p)) \cap P| - 1$.*

In order to prove Theorem 9.2 we use the following auxiliary lemma.

Lemma 9.1. *Let p_0 be a point in the plane and let $k \in \mathbb{N}$. Then there are at most $6k$ points Q such that p_0 is among the k closest points for each of the points $q \in Q$.*

Proof. We first establish the somewhat simpler claim that there are at most 6 points $p_1, \dots, p_t$ with $t \leq 6$ such that for each p_i no point is closer to p_i than p_0, that is, p_0 is the closest point to each p_i. Assume without loss of generality that p_1 is closest to p_0 and consider the infinite ray r_1 starting in p_0 in the direction of p_1. Let p_2 be the next point to p_1 in the clockwise cyclic order of $p_1, \dots, p_t$ around p_0, as illustrated in Figure 9.7. Without loss of generality we may assume that p_2 is such that the disk D_2 centered at p_2 with radius $d(p_2, p_0)$ touches p_1. If this is not the case, we rotate p_2 around p_0 counter-clockwisely until it touches p_2. By the

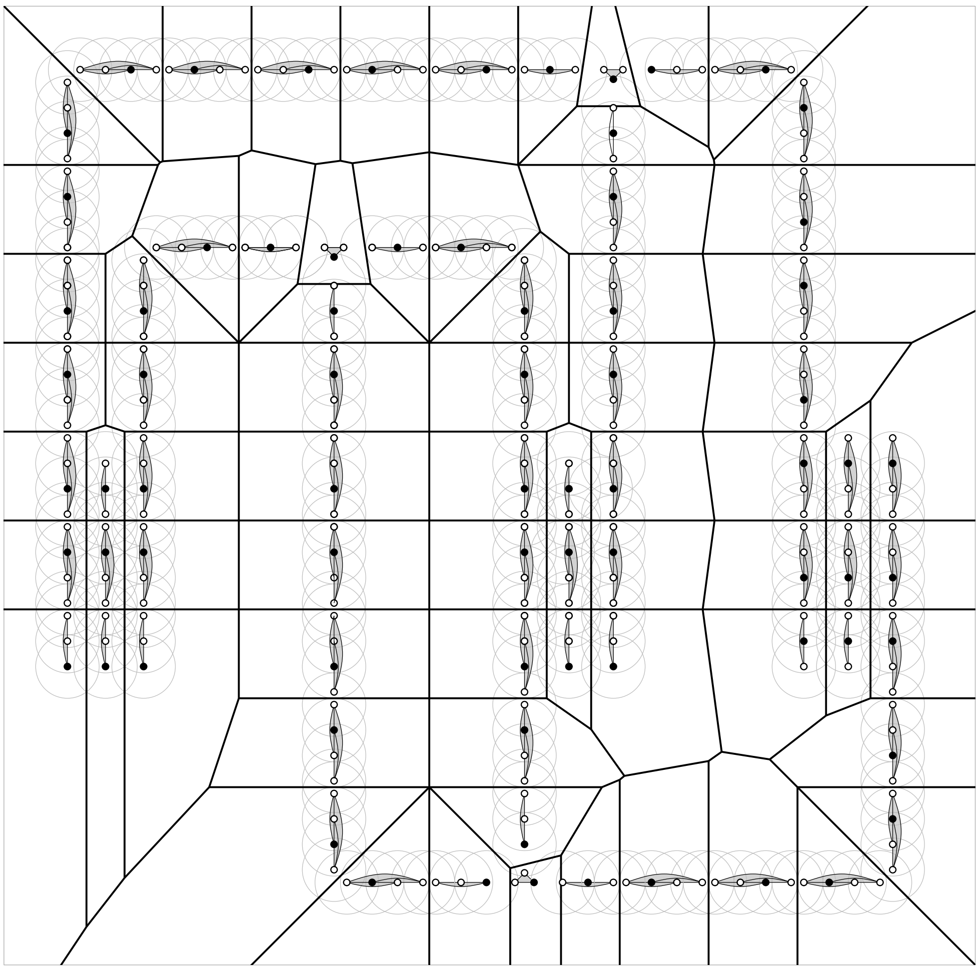


Figure 9.6: Sample reduction of the 3-SAT formula $(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_3 \vee x_4) \wedge (\overline{x}_2 \vee \overline{x}_3 \vee \overline{x}_4)$. The black points constitute a perfect ϱ-set corresponding to the assignment $x_1 = \text{true}, x_2 = \text{true}, x_3 = \text{true}, x_4 = false$.

choice of p_2 there will be no point in the disk around p_2 in its new position. If D_2 touches both p_0 and p_1, then its center must be on the bisector b of p_1 and p_2. Since it must also be outside D_1, the disk centered at p_1 with radius equal to $d(p_0, p_1)$, p_2 cannot be closer to p_1 on b than the intersection x of b and the boundary of D_1. Since p_0, p_1 and x form the corners of a equilateral triangle, the angle between p_0p_1 and p_0p_2 must be at least 60 degrees. By repeating the argument, it is clear that the angle between p_0p_1 and p_0p_i increases by 60 degrees for each $i = 2, \dots, t$. After at most 6 steps, this angle is at least 360 degrees. Hence there are at most 6 points with the desired property. Since we can put at most $k - 1$ additional points in each of the disks, the claim of the lemma holds. $\square$

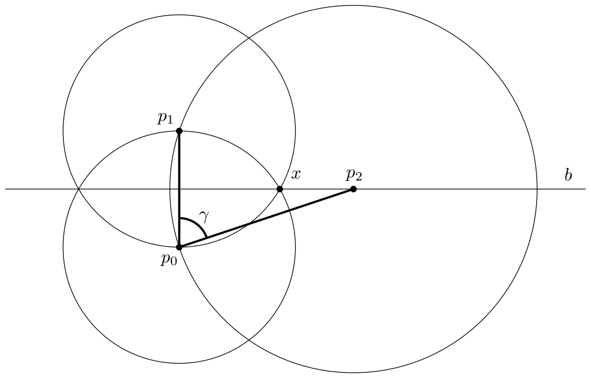


Figure 9.7: Illustration for the proof of Lemma 9.1

Now we are ready to prove Theorem 9.2.

Proof of Theorem 9.2. Let H be the graph on the set of points such that pq is a (directed) edge if and only if $q \in B(p, \varrho(p))$. The graph H contains an independent set of size s if and only if G contains a ϱ-set of this size. Each independent set in H corresponds to a ϱ-set in G since each point in H is connected to all points that are closer than $\varrho(p)$ and it is connected to all points q such that p is in the $\varrho(q)$ disk around q. On the other hand, each ϱ-set in G induces an independent set due to this construction.

By choice of ϱ, each vertex has out-degree bounded by $k = \max_{p \in P} |B(p,r) \cap P| - 1$ for any value of α. There is an ingoing edge from q into p if and only if p is among the k closest neighbors of q. By Lemma 9.1 there are at most $6k$ points such that p is among the closest k points for each of these points. Hence, the in-degree of each vertex is bounded by $6k$. In total, each vertex has degree at most $7k$. Hence, by a result due to Halldórsson and Radhakrishnan [HR97] we can approximate the maximum size of an independent set by a factor of $(7k + 2)/3$. The algorithm greedily chooses the minimum degree vertex in each step and can be implemented to run in time $\mathcal{O}(kn)$, given the graph H.

In order to compute H we locate the points in a closed disk by a circular range query in $\mathcal{O}(\log n + k)$ time using $\mathcal{O}(n \log^5 n (\log\log n)^2)$ preprocessing time [CCPY86]. Hence, the total running time is $\mathcal{O}(kn + n \log^5 n (\log\log n)^2)$. □

Based on this approximation, we heuristically compute a ϱ-set Q balancing both the size of Q and the local coverage of the Voronoi clustering induced by Q as follows. For $p \in P$ let $\widetilde{m}(p)$ denote the number of edges whose endpoints are both contained in $B(p, \varrho(p)/2)$ and let $\widetilde{n}(p)$ denote the number of points in $B(p, \varrho(p))$. We can use these values to compute an estimate of the local coverage as summarized in the following lemma.

Lemma 9.2. *Let Q be an inclusion-maximal ϱ-set and let $\alpha = 0$. Further, let $H = (Q, F)$ be the generalization obtained from $G = (P, E)$ by the Voronoi-mapping ν. Then the value*

$$\min_{q \in Q} \left\{ \frac{2\widetilde{m}(q)}{\widetilde{n}(q)(\widetilde{n}(q) - 1)} \right\}$$

is a lower bound for $\operatorname{lcov}(H, \nu)$.

Proof. For $\alpha = 0$ we have $\varrho \equiv r$. Whenever p is chosen as a cluster center in Q, the points in $B(p, r/2)$ are closer to p than to any other point in Q, since the closest point to p in Q has distance to p at least r. Hence, the edges in $B(p, r/2)$ are intra-cluster edges of C_p. On the other hand, the number of points in each of the clusters is bounded by $\widetilde{n}(p)$ whenever $\alpha = 0$ and Q is an inclusion-maximal ϱ-set. To see this, consider any vertex q that is not contained in $B(p, r)$, but closer to p than to any other cluster center. Then q is contained in none of the disks centered in the cluster centers and, thus, q must be a cluster center itself, since Q is inclusion-maximal. Hence, the claim holds. □

Based on Lemma 9.2 we propose a heuristic, called Greedy Weight Heuristic, that operates as follows. First we compute an estimate of $2\widetilde{m}(q)/(\widetilde{n}(q)(\widetilde{n}(q) - 1))$ for each $p \in P$ since computing the exact value involves complicated algorithms and data structures. Subsequently, we sort the points according to these estimates in $\mathcal{O}(n \log n)$ time and iteratively consider the points in this order. If the current vertex is not covered by the ϱ-disk of a previous vertex, then it is chosen for the ϱ-set, otherwise it is discarded.

Instead of computing $\widetilde{m}(p)$ and $\widetilde{n}(p)$ exactly, we estimate these numbers by counting the number of vertices and edges in the bounding boxes of the disks $B(p, \varrho(p)/2)$. To count the number of edges we use a 4-dimensional range searching query on a data structure containing tuples of points corresponding to edges in E with query time $\mathcal{O}(\log^3 m)$ [Cha88]. We use the 2-dimensional counterpart to locate points. Further, we use a data structure for dynamic nearest neighbor queries with $\mathcal{O}(\log^2 n)$ query time [BS80], into which we insert the selected points to decide whether the current point is covered by a previously selected point. The total running time is $\mathcal{O}((n + m) \log^3 m + n \log^2 n)$.

9.3 Minimizing Edge-Clutter

In order to reduce the clutter resulting from an excess of edges in certain areas we must filter out some of the edges without destroying the visual appearance of the graph. The total length of the edges seems to be a good measure for the clutteredness of the graph since it is proportional to the ink used for the drawing. While a minimum spanning tree will minimize this quantity, it is unlikely to preserve the visual appearance of the graph. We therefore require that monotone tendencies of the edges are preserved in order to best maintain the mental map of the adjacencies between vertices of the graph. This also motivated from a recent work by Huang et al. [HEH09], whose controlled user experiments seem to suggest that geodesic paths are more likely to be explored when reading a graph drawing.

Let ℓ be a line in the plane and let $S = (p_1, \ldots, p_k)$ be a sequence of points. We say that S is *ℓ-monotone* if the order of the orthogonal projections of $p_1, \ldots, p_k$ onto ℓ is the same as the order of the points in S. Let $G = (P, E)$ be a geometric graph and let (H, φ) be a generalization of G such that $H = (P, F)$, that is, $F \subseteq E$. We say that H is a *monotone generalization of* G if for every edge $e \in E$ with endpoints p and q there is a p-q-path π_e in H such that π_e is ℓ_e-monotone, where ℓ_e is the line defined by the endpoints of e. Given $G = (P, E)$ the Shortest Geodesic Subgraph (SGS) problem asks for a monotone generalization H of G minimizing the total length of H.

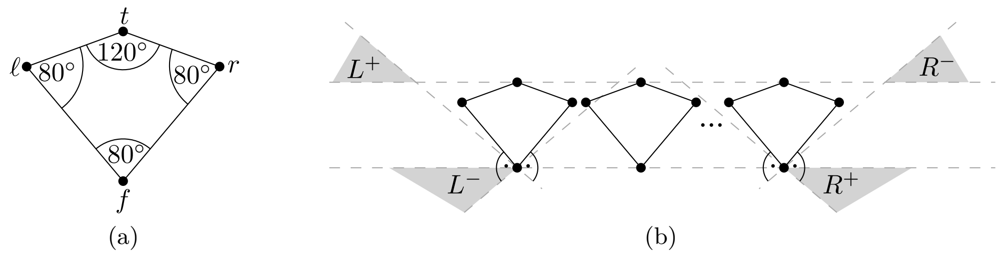


Figure 9.8: Overview of the reduction from 3-SAT to SHORTEST GEODESIC SUBGRAPH. A kite with foot point f, top point t and left and right points r and ℓ (a), and the arrangement of the kites in the reduction with the corresponding regions for clause vertices (b).

Problem SHORTEST GEODESIC SUBGRAPH (SGS)

Instance: Geometric graph $G = (P, E)$, $\delta \in \mathbb{R}_0^+$

Solution: Geometric graph $H = (P, F)$ such that $F \subseteq E$ and such that H contains a *monotone path* for each edge $e \in E$

Goal: minimize total length of H

First, we show that SHORTEST GEODESIC SUBGRAPH is NP-hard.

Theorem 9.3. SHORTEST GEODESIC SUBGRAPH *is NP-hard.*

Proof. We reduce from monotone 3-SAT, a variant of 3-SAT where each clause contains either only positive or only negative literals. Monotone 3-SAT is NP-complete [GJ79]. Let φ be an instance of monotone 3-SAT with variables $x_1, \ldots, x_n$ and clauses $C_1, \ldots, C_m$. We construct the following instance G_φ of SHORTEST GEODESIC SUBGRAPH. For each variable x we create a kite consisting of vertices ℓ, r, t and f as shown in Figure 9.8a. Note that the angles at f, ℓ and r are strictly less than 90°, and the angle at t is strictly more than 90°. The two edges incident to the top vertex t are called *top edges*, the edges $f\ell$ and fr are called the left and right *side edges*, respectively. We place the kites so that their foot points lie evenly spaced on the x-axis and the kites are disjoint. Denote by s_ℓ and s_r the slopes of the left and right side edges of a kite, respectively. The region R^+ is the region below the x-axis and to the right of the line through the bottom point of the rightmost kite with slope $-1/s_r$ (that is, it is perpendicular to the right sides of the kites). Further, we denote by L^+ the region that is above the horizontal line defined by the topmost points of the kites and to the left of the line with slope $-1/s_r$ through the foot point of the leftmost kite. We define R^- and L^- analogously, with s_r replaced by s_ℓ.

It follows immediately from the construction that a path that is monotone in the direction from a point in R^+ to a point in L^+ may not contain any right edge of a kite as this would imply a turn of more than 90°, which is not monotone. Analogously, monotone paths from L^- to R^- may not contain left edges of kites. In our reduction the kites will play the role of variables, and edges from R^+ to L^+ (from L^- to R^-) will play the role of clauses with only positive (only negative) literals.

For each clause C_i consisting of only positive literals, we add a *clause vertex* c_i^1 into R^+ and a clause vertex c_i^2 in L^+. We add *connector edges* that connect c_i^1 to the foot points of all

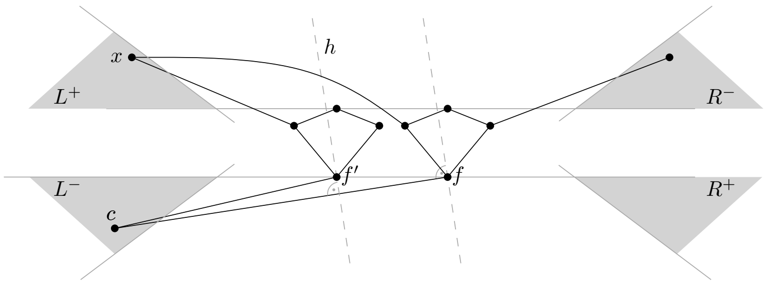


Figure 9.9: Illustration for the proof of the first claim. There is no monotone path in G_φ replacing the edge cf since every path avoiding this edge first visits a footpoint f' of a kite before it visits a point $x \in L^+$ whose orthogonal projection onto the line defined by c and f is to the left of the projection of f'.

kites that correspond to variables that occur in C_i and that connect c_i^2 to all the left points of kites that correspond to variables that occur in C_i. Finally, we add the *clause edge* $c_i^1 c_i^2$. We treat the clauses consisting of only negative literals analogously, except that we place the new vertices in L^- and R^-, respectively, and we connect the new vertices in R^- to the right kite points instead to the left.

This completes our construction, and we claim that an optimal solution of this instance allows us to decide whether the initial formula φ was satisfiable. We will make this more precise in the following. A subset of edges of G_φ is called *tight* if it contains both top edges of each kite, all connector edges, and exactly one of the two side edges of each kite. We now claim the following.

Claim 9.1. Any feasible solution contains a tight edge set.

Proof of Claim. First note that the top vertex of each kite is incident to only two edges, hence at least one of them must be in any feasible solution. However, the left edge is not monotone in the direction of the right edge and vice versa. Hence, a feasible solution necessarily contains both of them.

Next, we show that all edges from clause vertices in L^- or R^+ to foot vertices of kites must be contained in every solution. Let c be a vertex in L^- (the case c in R^+ is symmetric) and let f a foot point of a kite that is adjacent to c. We now consider the paths from c to f that avoid the edge cf in our graph. Since G contains no edge connecting two vertices of different kites, any path from c to f that avoids cf must contain at least one vertex $x \neq c$ that is in one of the four regions L^+, L^-, R^-, and R^+. Note that, by construction of the region L^-, the line orthogonal to cf is at least as steep as the left side of any kite, and hence the points in R^- and in R^+ lie to the right of the line that is orthogonal to cf through f, and thus are not part of any monotone connection from c to f as illustrated in Figure 9.9. Now assume that x is in L^+ or L^-, and denote by f' the first vertex after c on a cf-path that avoids the edge cf. The regions L^+ and L^- lie to the left of the line h that is orthogonal to cf through the foot point of the leftmost kite. Therefore both the edge cf' and the subpath from x to f must cross this line, and hence project to the same point on the line segment from c to f. This shows that the path is not monotone, and hence cf must be contained in any feasible solution.

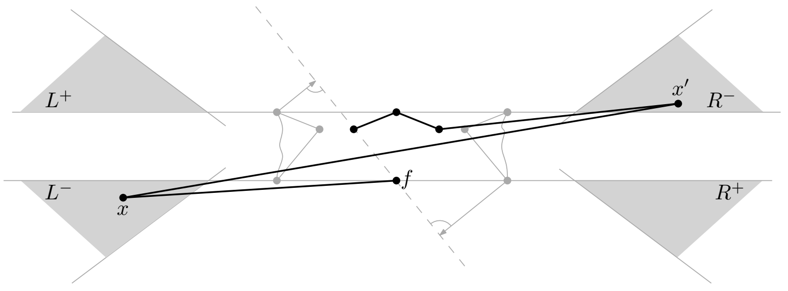


Figure 9.10: Illustration for the proof of the first claim. In each of the kites at least one of the side edges must be present. Otherwise, any replacing path for $f\ell$ must use an edge xx' that is not monotone with respect to $f\ell$.

Next, consider a vertex c in L^+ and a corresponding edge $c\ell$ to the left vertex of a kite (again the case c in R^- and edge cr where r is the right vertex of a kite is symmetric). As before, every path from c to ℓ avoiding $c\ell$ must contain a vertex $x \neq c$ belonging to one of the four regions. Again, the regions R^- and R^+ are to the right of the line orthogonal to $c\ell$ through ℓ, and can thus not be contained in a monotone $c\ell$-path. Hence, we can assume that x is in L^+ or L^-. Let ℓ' be the first vertex after c on a $c\ell$-path that avoids the edge $c\ell$. If x is in L^-, consider the line orthogonal to $c\ell$ through the foot point of the leftmost kite. By construction this line is at least as steep as the right side of a kite, and hence the region R^+ is to its left. Since any path from c to x must contain a foot vertex of a kite, both the subpath from c to x and the subpath from x to f must cross this line, and thus project to the same point on the edge $c\ell$. Hence the path would not be monotone and we can assume that x is in L^+. Considering the line orthogonal to $c\ell$ through the left point of the leftmost kite as above rules out the existence of such a monotone path.

It remains to show that at least one side edge of each kite must be in any feasible solution. Let f be the foot point of a kite K with left point ℓ, right point r and top point t. We show that G does not contain a monotone $f\ell$-path that avoids both $f\ell$ and fr. First observe that all foot points of kites to the right of K project before f on the line through f and ℓ, directed from f to ℓ, and hence cannot be contained in a monotone $f\ell$-path. Similarly, all non-foot vertices of kites to the left of K project behind ℓ on this line, and hence are also not contained in monotone $f\ell$-paths. The points in R^+ and all points in L^+ can be ruled out similarly. Since a monotone $f\ell$-path needs to contain an edge that connects a vertex whose y-coordinate is at most the y-coordinate of f to a vertex whose y-coordinate is at least the y-coordinate of ℓ, and we cannot use any edge of a kite, the only option is that it uses an edge from a vertex x in L^- to a vertex x' in R^- as illustrated in Figure 9.10. However, the line orthogonal to such an edge is steeper than the edge $f\ell$, and hence xx' is not monotone with respect to $f\ell$. This completes the proof of the claim.

Note that the size, as well as the total length, is the same for all tight edge sets, and hence this size forms a lower bound for the size of a geodesic subgraph. We claim that this bound can be met if and only if φ is satisfiable.

Claim 9.2. There exists a tight set that is feasible if and only if φ is satisfiable.

Proof of claim. Note that a tight set is completely specified by giving for each kite the information whether its left or right edge is contained in the set.

Assume that φ is satisfiable and take a satisfying assignment. We construct a tight set E' by taking the left side of a kite if and only if the corresponding variable has the value *true* in the assignment. We now argue that the corresponding set is feasible. The only edges for which we have to check the existence of a monotone replacement path are the clause edges. Let $c_i^1 c_i^2$ be a clause edge with c_i^1 in R^+ and c_i^2 in L^+. The edge $c_i^1 c_i^2$ by construction corresponds to a clause C_i with only positive literals. Let x_j be a satisfied literal (and thus a satisfied variable) in C_i, and let K_j denote the corresponding kite with foot point f and left point ℓ. By construction E' contains the edges $c_i^1 f, f\ell$ and ℓc_i^2, which together form a monotone $c_i^1 c_i^2$-path. The argument for a clause edge $c_i^1 c_i^2$ with c_i^1 in L^- and c_i^2 in R^-, which corresponds to a clause with only negative literals, is analogous. This proves that the tight set E' is feasible.

Conversely, assume that E' is a feasible tight set. We construct a truth assignment by setting a variable to true if and only if the left edge of the corresponding kite is in E'. Now consider a clause C_i containing the variables x_u, x_v and x_w as positive literals (the case of only negative literals is symmetric). If C_i is not satisfied by our assignment then E' contains none of the left edges of the three kites corresponding to x_u, x_v and x_w. However, by definition the edge set must contain a monotone $c_i^1 c_i^2$-path. Such a path may not use any right edge of any kite as this would not be monotone. Hence it necessarily contains a left edge of some kite since E' does not contain any of the clause edges. This implies that any monotone path must first visit the foot point of one of the kites corresponding to x_u, x_v, x_w, then pass on to a vertex $x \neq c_i^1$ in L^- or R^+ and from there to the foot point of another kite. By construction all points in R^+ lie to the right of the line that is orthogonal to $c_i^1 c_i^2$ through the foot of the rightmost kite. This excludes the case that x is in R^+. Similarly, the line orthogonal to $c_i^1 c_i^2$ through the foot of the leftmost kite separates the points in R^- from the foot points of all kites. Hence the edges from x to its two incident foot points both cross this line and hence the path is not monotone. This is a contradiction, and hence E' must contain the left edge of at least one of the kites corresponding to x_u, x_v and x_w, thus implying that C_i is satisfied. This proves the claim.

Note that the size L is the same for all tight edge sets. The first claim shows that any geodesic subgraph has length at least L. And thus, the second claim implies that φ is satisfiable if and only if G_φ admits a geodesic subgraph of length at most L. Since the construction can easily be performed in polynomial time this concludes the proof. $\square$

As we have seen, the restriction to edges from the input graph makes it difficult to construct short monotone subgraphs. One possibility is thus to drop this constraint and to allow arbitrary edges. Additionally, we would like to control the distance of the monotone path π_e and the edge it is approximating in terms of monotonicity. This is motivated by the observation that the shortest monotone generalization of a clique, whose vertices are arranged equidistantly on a circle, is given by the convex hull of the point set. Given a line segment s with length ℓ_s and a point p with distance d_p from s we call the ratio d_p/ℓ_s the *drift of p from s*. The *drift of a path π_e* with endpoints pq is defined as the maximum drift of any point on π_e from the segment pq. Given a geometric graph $G = (P, E)$ and a non-negative real number $\delta \in \mathbb{R}_0^+$ the Sparse Geodesic Network (SGN) problem asks for a geometric graph $H = (P, F)$ with minimum total length such that for each edge e in E there is an ℓ_e-monotone path π_e with drift at most δ, where ℓ_e denotes the line defined by the endpoints of e.

Problem Sparse Geodesic Network (SGN)

Instance: Geometric graph $G = (P, E)$, $\delta \in \mathbb{R}_0^+$

Solution: Geometric graph $H = (P, F)$ such that H contains a *geodesic path* for each edge $e \in E$ whose vertices are at distance at most $\delta \cdot |e|$ from the straight line e

Goal: minimize $|F|$

We show the following.

Lemma 9.3. *Given a (complete) geometric graph $G = (P, E)$, the Delaunay graph $\mathcal{D}(P)$ contains for each edge $e \in E$ an ℓ_e-monotone path π_e with drift at most* 1/2.

Proof. Let P be a set of points and let $p, q \in P$. Without loss of generality we assume that p and q are on the x-axis such that $x(p) < x(q)$. According to Dobkin et al. [DFS90] we can construct an x-monotone path in the Delaunay graph $\mathcal{D}(P)$ of P as follows. Let $\mathcal{V}(P)$ denote the Voronoi diagram of P and let $p_1, \ldots, p_k$ be the ordered points corresponding to the Voronoi cells that are traversed when following the line from p to q. Then the path $p, p_1, \ldots, p_k, q$ is an x-monotone path in the Delaunay graph. Further, all points p_i are contained within the disk with radius $d(p, q)/2$ centered in the midpoint of the segment pq. Hence, the drift is at most 1/2. □

Although the Delaunay graph seems to be well suited to represent monotone tendencies, this result also shows the limitations of allowing arbitrary edges. In the following we therefore focus on subgraphs of the original graph and describe a greedy heuristic for computing a monotone generalization with bounded drift δ and short total length, which we call Monotone Drift Heuristic. Given a geometric graph $G = (P, E)$ and a maximal drift δ we sort the edges of G with respect to increasing length in $\mathcal{O}(m \log m)$ time. Then we consider the edges $e_1, \ldots, e_m$ in this order and iteratively construct a sequence of graphs $H_0, H_1, \ldots, H_m$, where $H_0 = (P, \emptyset)$. We insert the edge e_i into H_{i-1} whenever there is no ℓ_{e_i}-monotone path with drift at most δ in H_{i-1}. This can be tested by performing a modified depth-first search exploring only monotone subpaths in $\mathcal{O}(n + m)$ time. Hence, the total running time of this approach is $\mathcal{O}(nm + m^2)$.

9.4 Vertex-Edge-Clutter

Vertex-edge-clutter is the most complicated type of clutter since it involves both vertices and edges and the selection of these features cannot be handled independently as in the previous sections. On the other hand, this type of clutter may be considered as the least annoying type of clutter. While vertex-edge clutter is caused by edges that are close to a vertex resulting in the difficulty to determine correct incidences,the human perception is rather good at determining whether a line passes a disk through the center or not. For instance, it is easy to see that the leftmost line in Figure 9.11a is not incident to the vertex although it crosses the vertex. Additionally, the human perception is also good at determining whether a line has a bend or not, which is illustrated in Figure 9.11a.

Hence, as long as there is neither vertex-clutter nor edge-clutter and as long as no pair of edges incident to a common vertex form a 180°-angle, we will be able to unambiguously tell whether an edge is incident to a vertex or not. In order to attack vertex-edge clutter we therefore propose the following optimization problem. For a pair of edges incident to a common vertex p we define the *angular straight-line deviation* as the smaller of the two angles that is enclosed by the lines defined by the two edges, respectively. The angular straight-line deviation of p is then defined as the minimum angular straight-line deviation over all pairs of edges incident to p, as illustrated in Figure 9.11b. The angular straight-line deviation of a geometric graph G is the minimum angular straight-line deviation over all vertices of G. Note, that the angular straight-line deviation is maximized if all angles are close to a right angle. Given a geometric graph $G = (P, E)$ and a non-negative value $r \in \mathbb{R}^+$, the Optimal Angle Adjustment problem is to find a new position for each vertex p inside $B(p, r)$ minimizing the angular straight-line deviation of the resulting geometric graph.

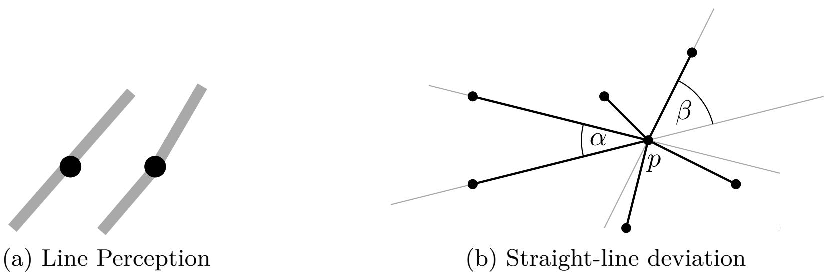
Problem Optimal Angle Adjustment

Instance: Geometric graph $G = (P, E)$, $r \in \mathbb{R}_0^+$

Solution: Geometric graph $H = (Q, F)$ and a mapping $f : P \to Q$ such that $d(p, f(p)) \leq r$ and such that $f(p)f(q) \in F$ if and only if $pq \in E$

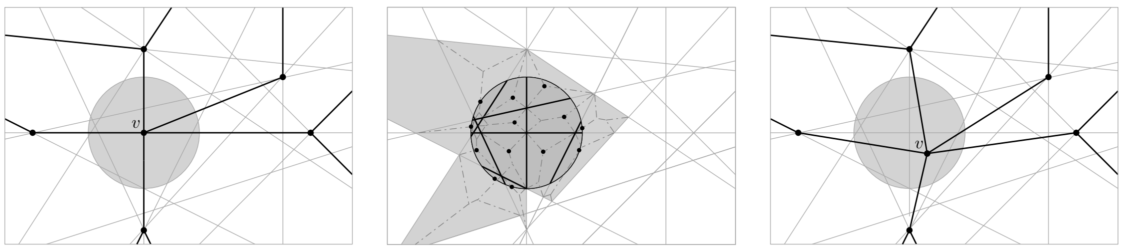
Goal: maximize the angular straight-line deviation of H

Note that this problem differs considerably from the problem of maximizing the angular resolution of a graph, defined as the minimum angle over all pairs of adjacent edges. The optimal angular resolution of a star-shaped graph with an odd number n of vertices, for instance, will result in zero straight-line deviation, while it is obvious that the optimal straight-line deviation is positive. We tackle the Optimal Angle Adjustment problem by maximizing the vertices' distances from the lines defined by the edges incident to their neighbors. Let $G = (P, E)$ be a geometric graph and let $v \in P$ be a vertex. Let $N(v)$ denote its neighbors in G. Further, let $E(v)$ denote the edges incident to v and let $F(v)$ denote the set of edges incident to the vertices in $N(v)$ but not to v. By moving v we change the angles formed by pairs of edges in $E(v)$ as well as the angles formed by pairs of edges (e, f) such that $e \in E$ and $f \in F$, respectively. Let $L_F(v)$ be the set of lines defined by the edges in



(a) Line Perception (b) Straight-line deviation

Figure 9.11: Figure (a) shows two examples where we can clearly distinguish whether an edge in incident to a vertex. On the left side, it is clearly visible that the vertex is not incident to the edge although it intersects the edge. On the right side, it is clear that the vertex is incident to the edges since are not aligned. Figure (b) shows an illustration of the angular straight-line deviation of a vertex p. In the drawing the angular straight-line deviation is defined by the angle α.



(a) (b) (c)

Figure 9.12: Illustration for the proof of Theorem 9.4. (a) A vertex v and its neighbors as well as the arrangement of lines induced by the respective edges in $L_E(v)$ and $L_F(v)$. (b) Intersection of the circle with projections of the graphs $\mathcal{G}_c$ (dashed) and locally optimal positions (black dots) in the faces. (c) Globally optimal position and resulting new drawing.

$F(v)$ and let $L_E(v)$ be the set of lines defined by all pairs of vertices in $N(v)$. A vertex v along with the lines defined by the edges in $L_E(v)$ and $L_F(v)$ is illustrated in Figure 9.12a. Note, that there will be an angle of 180 degrees involving an edge incident to v if and only if v is placed on one of the lines in $L_E(v) \cup L_F(v)$. Given $p \in \mathbb{R}^2$ we denote by $\mu_v(p)$ the minimum distance of p to the lines in $L_E(v) \cup L_F(v)$. We prove the following.

Theorem 9.4. *Given a graph $G = (P, E)$, a vertex $v \in P$ and a positive radius $r \in \mathbb{R}^+$ we can compute a new position p^* for v in $B(v, r)$ such that $\mu_v(p^*) > 0$ and such that p^* maximizes $\mu_v(p)$ over all $p \in B(v, r)$ in $\mathcal{O}(t^3\alpha(t))$ time where $t = \min\{\Delta^2, m\}$, Δ denotes the maximum degree of G and $\alpha(\cdot)$ denotes the inverse Ackermann function.*

Proof. First, we compute the set of edges $L_F(v)'$ incident to v's neighbors, but not to v, that intersect $B(v, r)$ as well as the set of lines $L_E(v)'$ defined by all pairs of v's neighbors intersecting $B(v, r)$. Let $L = L_E(v)' \cup L_F(v)'$. We compute the arrangement of lines in L in $\mathcal{O}(|L|^2)$ time. Over each of the resulting faces C we compute the lower envelope of the hyperplanes defining the distance to the boundaries of the faces and project the graph $\mathcal{G}_C$ defined by the resulting 3-dimensional polytope onto the plane. This is illustrated in Figure 9.12b.

The lower envelope of a set of n hyperplanes can be computed in $\mathcal{O}(n^2\alpha(n))$ time where $\alpha(\cdot)$ denotes the inverse of the Ackermann function [EGS89]. Hence the lower envelopes can be computed in time $\mathcal{O}(|L|^2\alpha(|L|))$ for each face, resulting in a total complexity of $\mathcal{O}(|L|^3\alpha(|L|))$. For each face C we inspect the vertices of $\mathcal{G}_C$ in $B(v, r)$ as well as its intersection with $B(v, c)$ and thus compute the point p^* maximizing μ_v in $B(v, r)$. Then we update the position of v as illustrated in Figure 9.12c. Since L is bounded by $\max\{\Delta^2, m\}$ we obtain the claimed time complexity. Further, since $r > 0$ and therefore $B(v, r)$ is non-degenerate, there must be a non-degenerate face in the arrangement containing a point p^* in its interior such that $\mu(p^*) > 0$. □

Using Theorem 9.4 we can incrementally compute a new position for each vertex v such that none of the edges incident to v encloses an angle of 180 degrees with any other edge. Since the angles between pairs of edges that are not incident to v are not affected by this operation, we can iteratively apply Theorem 9.4 to the vertices one after another to obtain a drawing with strictly positive angular straight-line deviation. At the same time this approach heuristically maximizes this deviation.

Note that we may assume that we apply the angle adjustment to a generalized graph whose complexity tends to be significantly lower than the complexity of the original graph, that is, both m and Δ should be considerably smaller.

9.5 Sample Generalizations

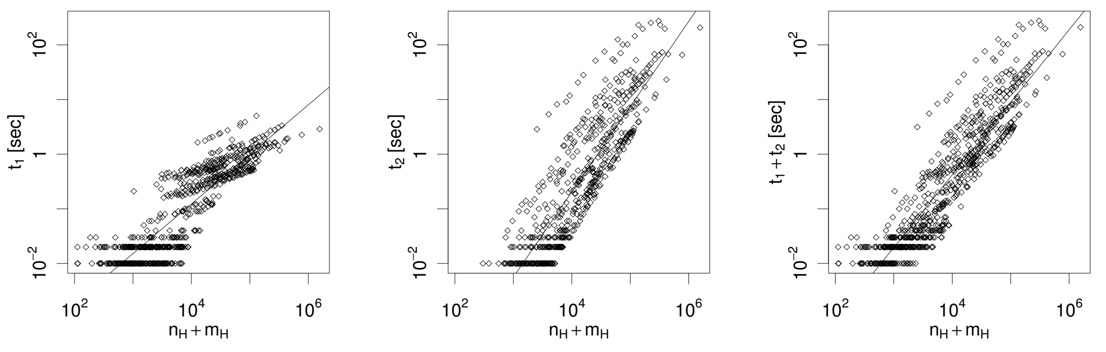
In order to evaluate the quality of the described heuristics and in order to obtain estimates for the running time, we implemented the described Greedy Weight Heuristic and Monotone Drift Heuristic in C++ using the BOOST library [boo] and the CGAL library [cga]. All generalizations were computed on a standard Intel Core 2 Duo processor running at 2.00 GHz with 2 GB RAM.

We performed our experiments on the benchmark set of graphs listed in Table 9.1 on page 226. These graphs have between 1,000 and 100,000 vertices and between 3,000 and 2,000,000 edges, respectively. The table lists, for each graph, an index that is used to identify the graphs in Figure 9.13 as well as its size. All but the graphs marked with $\star$ have been taken from the University of Florida sparse matrix collection [Dav94]. The graph `clique-planar` is a planar graph with an implanted clique. The graph `lunar-vis` is a LunarVis [GGW08] layout of a snapshot of the Internet graph at the autonomous systems level that has been taken from the data collected by the University of Oregon Routeviews Project [rou]. The graph `email` is a force-based visualization of the graph obtained from the e-mail communication at the faculty of informatics at the Karlsruhe Institute of Technology during a fixed amount of time. The graphs `osm_berlin` and `osm_isleofman` are street networks of Berlin, Germany, and Isle of Man, respectively, that have been extracted from the OpenStreetMap data [osm11]. The graphs from the University of Florida sparse matrix collection have additionally been layouted using the `sfdp` multi-scale force-based layouter from the graphviz library [EGK+03].

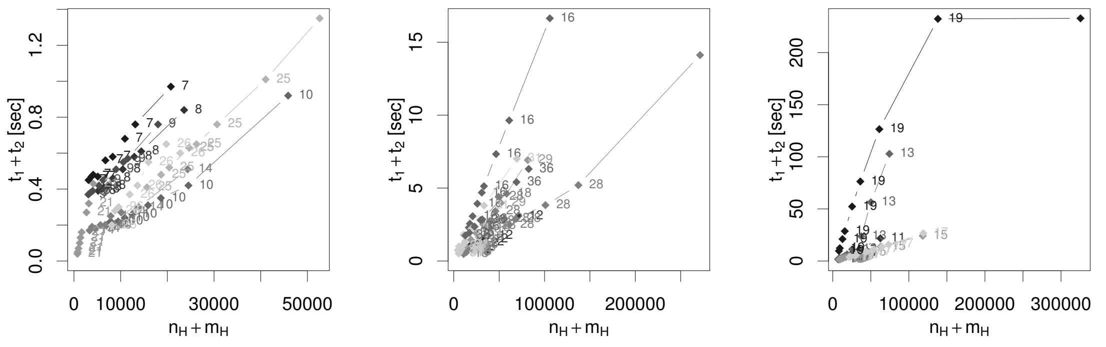
For each of the graphs listed in Table 9.1 as well as for both $\alpha = 0$ and $\alpha = 1$, we performed generalizations with 10 different radii $r_i = \Delta/n + (\Delta/\sqrt{n} - \Delta/N) \cdot i$ in the range $[\Delta/n, \Delta/\sqrt{n}]$ for $i = 0, \dots, 9$, where $\Delta := x_{\max} - x_{\min}$ is the width of the drawing. For each run, we measured the time t_1 of the Greedy Weight Heuristic and the time t_2 of the Monotone Drift Heuristic. Further, we collected the number of vertices n_H and the number of edges m_H of the resulting generalized graphs.

Even for the largest input graphs with several thousand vertices and over a million edges, the observed running times were less than 5 minutes. However, most of the running time is caused by the Monotone Drift Heuristic, which has a quadratic worst-case running time. For the Greedy Weight Heuristic, the observed running time was less than 5 seconds for all graphs.

Figure 9.13 shows the running time of the heuristics as a function of the size $n_H + m_H$ of the generalized graphs. Figure 9.13a shows the running time t_1 of the Greedy Weight Heuristic, Figure 9.13b shows the running time t_2 of the Monotone Drift Heuristic and Figure 9.13c shows the resulting combined running time. In order to display all data in one chart, we employed a log-log-scale plot for these figures. We added a line to each chart displaying the results of a linear regression, applied to the log-transformed data. That is,



(a) Vertex Clutter (b) Edge Clutter (c) Total



(d) $0.3 \leq t < 3$ (e) $3 \leq t < 20$ (f) > 20

Figure 9.13: Running times in seconds of the generalization heuristics with respect to the size of the generalization in a log-log-scale plot for $\alpha \in \{0, 1\}$ and drift $= 0.3$ (a)–(c) and running times for the individual graphs in a linear-scale plot for $\alpha = 1$ and drift $= 0.3$ (d)–(f). For reasons of clarity, Figure (a) contains only the graphs with $0.3 \leq t_1 + t_2 < 3$, Figure (b) contains only the graphs with $3 \leq t_1 + t_2 < 20$ and Figure (c) contains only the graphs with $t_1 + t_2 \geq 20$.

for $x = n_H + m_H$ and for each $y \in \{t_1, t_2, t_1 + t_2\}$ we computed a_y and b_y minimizing the linear least-squares function

$$\min_{a_y, b_y \in \mathbb{R}} \sum_{i=1}^{N} (\log y_i - (a_y \log x_i + b_y))^2$$

where x_i and y_i denote the measured sizes and running times of the single experiments for $i = 1, \ldots, N$, respectively. The results suggest that the running time of the Greedy Weight Heuristic $t_1 \approx e^{-10.4578} x^{0.9076}$ is approximated by a function that is slightly sub-linear in the size of the generalized graph and the running time of the Monotone Drift Heuristic $t_2 \approx e^{-15.95} x^{1.56}$ is approximated by a super-linear but sub-quadratic function in the size of the generalized graph. The combined running time is approximately $t_1 + t_2 \approx e^{-13.105} x^{1.328}$, which is super-linear.

Figures 9.13d–9.13f display the running times for $\alpha = 1$ for the individual graphs of our benchmark on a linear scale. Figure 9.13d contains all graphs for which $t_1 + t_2$ was at most 3 seconds, Figure 9.13e contains all graphs whose maximum running time was between 3 and 20 seconds and Figure 9.13f contains all graphs whose running time was more than 20 seconds. With only a few exceptions, such as `ex3sta1`, `TF16` and `conf5_4-8x8-05`, the running times seem to be slightly sub-linear or slightly super-linear in the size of the generalized graph. The results for $\alpha = 0$ are similar.

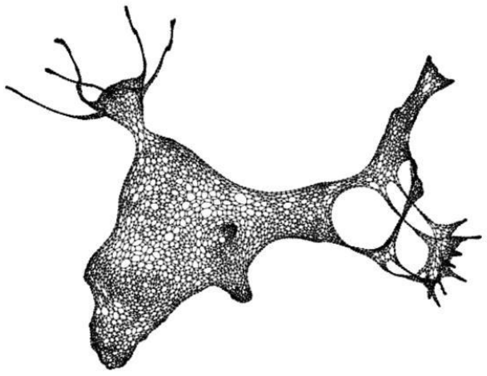
Next, we shortly discuss the generalized graphs. Figure 9.14 shows how the parameter α affects the generalization. While the sizes of the generalized graphs for $\alpha = 0$ and $\alpha = 1$ will, in general, differ considerably for a fixed radius r, we chose generalized graphs with roughly the same sizes in order to illustrate the effects of choosing $\alpha = 0$ and $\alpha = 1$, respectively. Figure 9.14 clearly shows that the generalizations with $\alpha = 1$ are better suited at preserving the distribution of the original point set. However, this is only achieved at the price of a higher resolution of the resulting drawing. On the other hand, the homogeneous distribution of the points resulting from $\alpha = 0$ does not seem to capture the geometric properties of the original very well. Indeed, it seems that setting $\alpha = 0$ is not well suited for most of the graphs we inspected due to this behavior. Therefore, all remaining generalization are performed with $\alpha = 1$ if not otherwise stated.

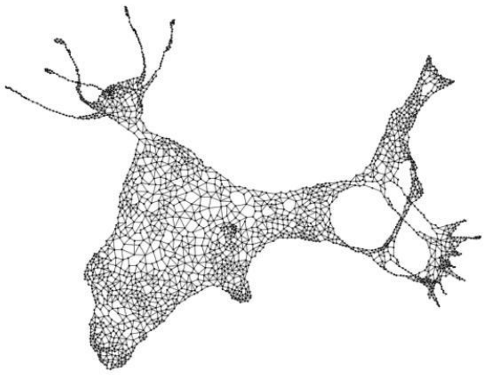
Figure 9.15 shows how the radius r and the drift δ impact the resulting generalizations for $\alpha = 1$. To illustrate the effects of δ we applied the Greedy Weight Heuristic and Monotone Drift Heuristic for different values of r and δ to a planar graph with an implanted clique whose vertices have been arranged equidistantly on a cycle. While none of the edges of the clique is distinctly perceivable in the original drawing, a higher drift helps remedying this without destroying the impression of a clique even without generalizing the vertex set, as can be seen in the first row of Figure 9.15. Note that the clique in the middle of the drawing remains a clique when setting $\delta = 0$ for all values of r. With increasing δ, the clique becomes much sparser but is still perceivable as a rather dense subgraph in the generalized graph for all radii.

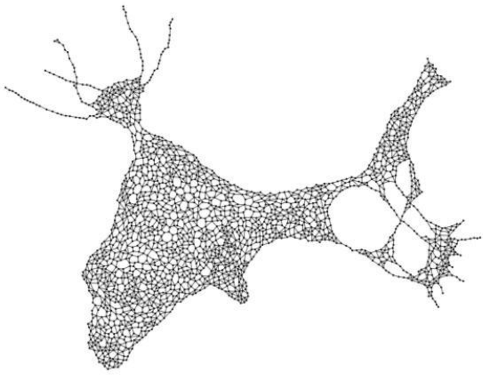
Finally, Figures 9.16–9.24 show some selected sample generalizations. First, we discuss how the heuristics perform on the graphs we used to illustrate the various types of clutter in Figure 9.1c. These graphs as well as the results of the heuristic generalization are displayed in Figures 9.16–9.18. Note that the displayed generalized graphs have only 2–5% of the vertices of the respective originals. Clearly, both vertex-clutter and edge-clutter can be significantly reduced without changing the main impression of the graph. However, two drawbacks of our approach are immediately obvious from these illustrations.

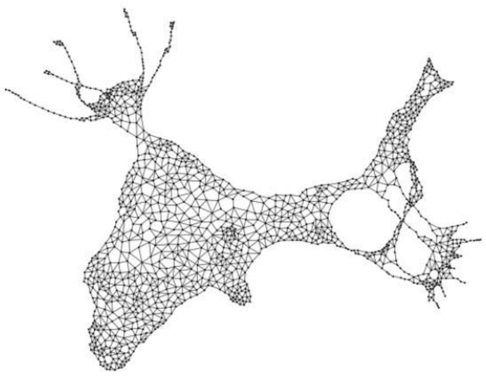
First, consider the graph `oix` and its generalization in Figures 9.17a and 9.17b, respectively. In the original, there are many edges between a few vertices on the left and the vertices in the bottom center. Apparently, these edges are mapped to only few monotone paths in the generalization, which changes the impression of the density of the edges. This could be tackled by emphasizing the edges in the generalization according to the number of edges that were mapped to it. As another approach to this problem we could try to approximate the geometric edge distribution. That is, similar to our approximation of the point-set distribution we could try to remove more edges from regions containing few edges and removing fewer edges in regions with many edges. In contrast to approximating the point-set distribution, however, it is not clear how to achieve this in a straightforward way since a single edge may cross both dense and less dense regions in the drawing.

Second, consider the graph `PDS10` and its generalization in Figures 9.18a and 9.18b, respectively. Clearly, the topmost vertices of the generalization show that our approach may


(a) original (n=7920, m=31680)


(b) generalization (n=2104, m=8585)


(d) generalization (n=2098, m=8188)


(c) generalization (n=1305, m=5636)

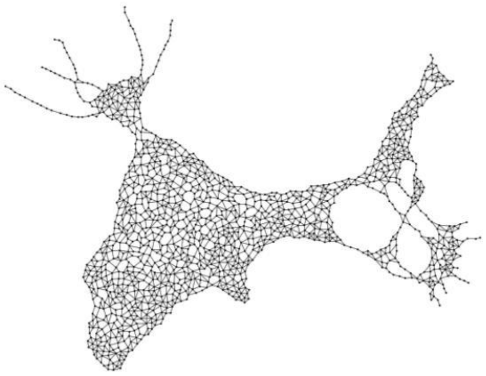

(e) generalization (n=1358, m=5678)

Figure 9.14: Effects of parameter α on the `commanche_dual` graph from the University of Florida sparse matrix collection [Dav94]. (a) Original, (b), (c) Generalization with $\alpha = 1$, (d), (e) Generalization with $\alpha = 0$.

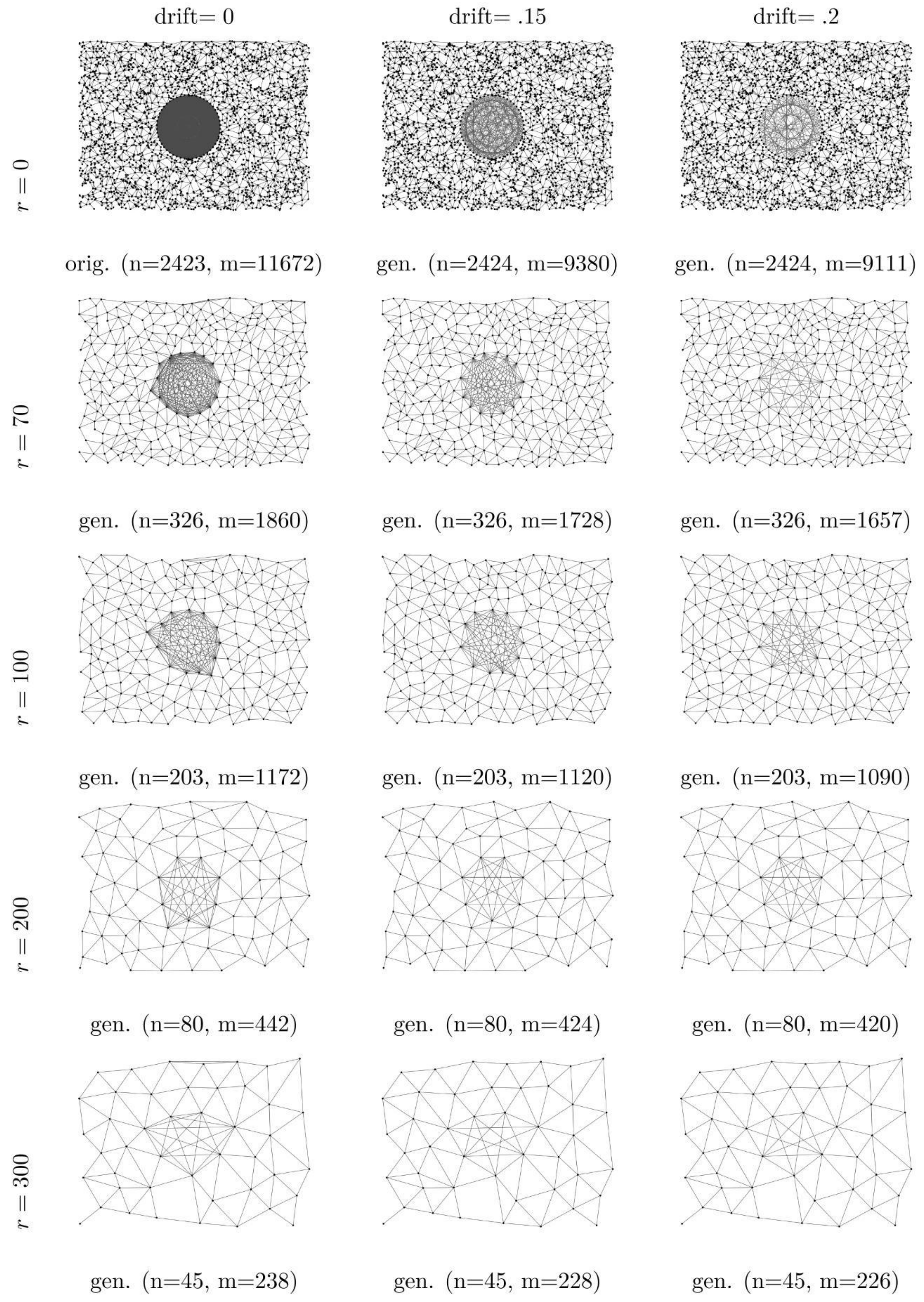


Figure 9.15: Effect of radius and drift on the graph `clique-planar`, a planar graph with an implanted clique. All generalizations with $\alpha = 1$.

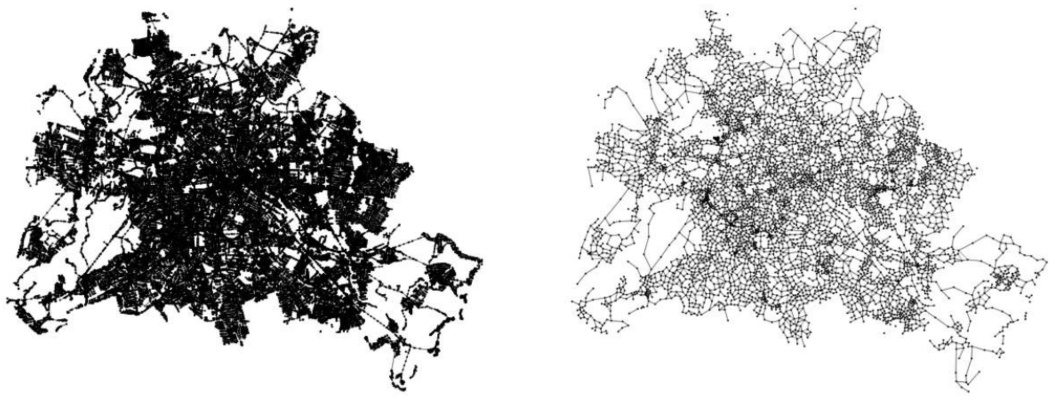

(a) original (n=106675, m=248390) (b) generalization (n=5649, m=17273)

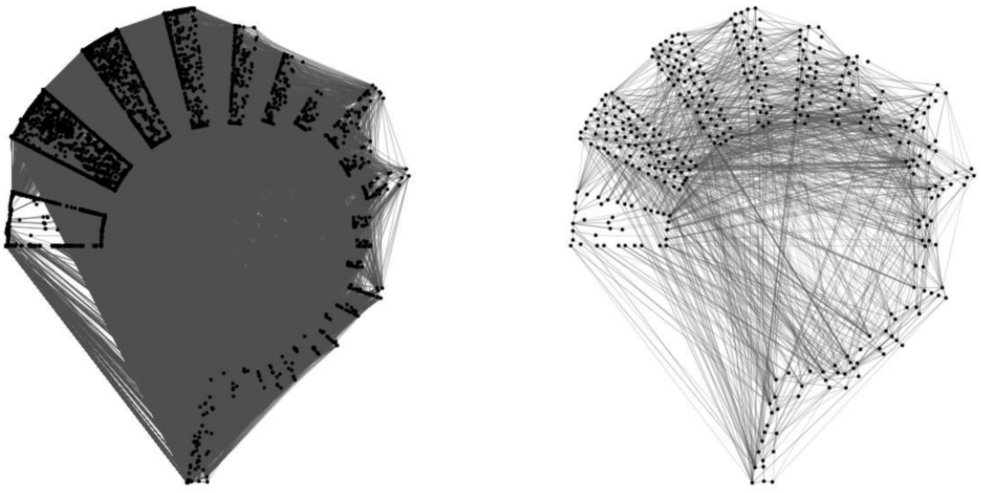
Figure 9.16: Streetmap data of Berlin [osm11] (`osm_berlin`).

create unwanted adjacencies. These adjacencies are the result of contracting vertices that are close to each other and working on the contracted edge set. While the edges are not false in the sense that each edge in the contraction corresponds to at least one edge of the original, these edges create the wrong visual impression. This problem could be approached by trying to approximate the features of the contracted vertex sets. For instance, the average degree of the contracted vertex sets will be roughly two for most of the problematic vertices in these figures, while the resulting degree in the generalization is larger.

The remaining figures serve as a further visual benchmark of the generalization heuristics. While the general (geometric) impression of the graphs are reasonably well maintained, some further issues for future research can be observed.

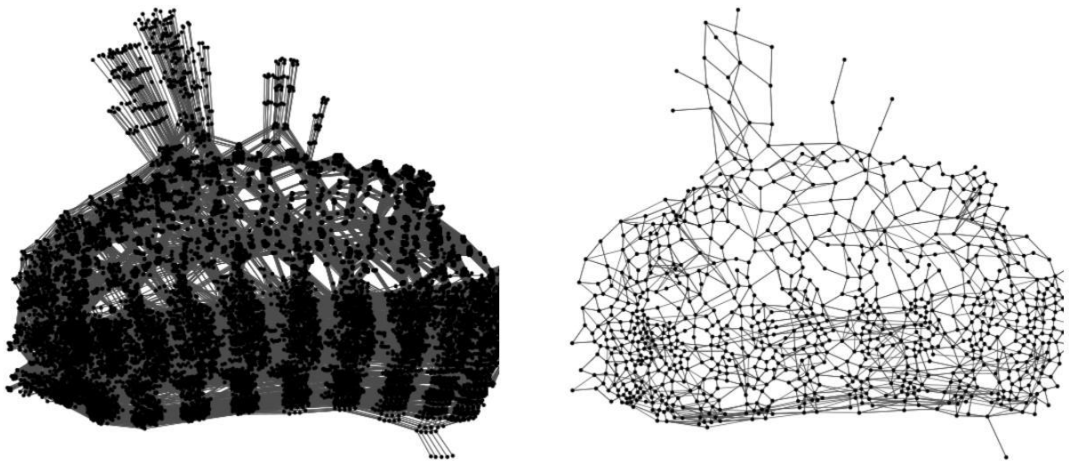
Consider, for instance the graph `ukerbe1_dual` and its generalization illustrated in Figure 9.22a and 9.22b, respectively. While the density of the point set and the size of the faces is maintained quite well, most of the faces are triangulated in the generalization, whereas most of the faces of the original contain four vertices. Further, consider the cube graph illustrated in Figure 9.23. The topological structure of the generalized graph is rather different from the original. Although the vertices contracted into single clusters are close to each other both geometrically and with respect to graph-distance, the cubic structure is not maintained. Again this may be remedied by approximating the features of the contracted vertices, such as average degree.

While the proposed heuristics do not solve the generalization problem in all its facets, especially with respect to the topological features of the graph, they seem to be well suited at maintaining the geometric impression of the originals and, thus, form good starting points for future research on this problem. In order to overcome the current difficulties, however, we must explicitly include topological features of the original graph into the generalization process.



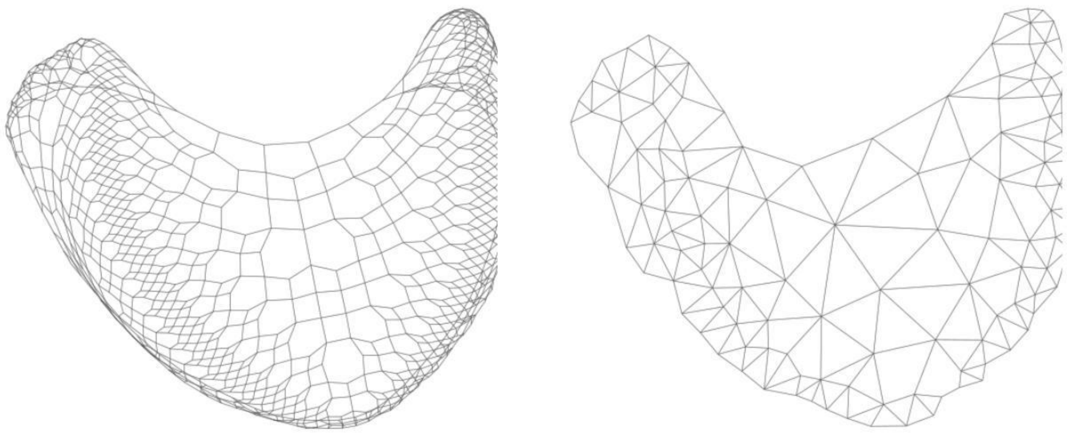
(a) original (n=17233, m=74436) (b) generalization (n=397, m=2134)

Figure 9.17: LunarVis Layout of the Internet at the autonomous-systems level [GGW08] (`lunar-vis`).



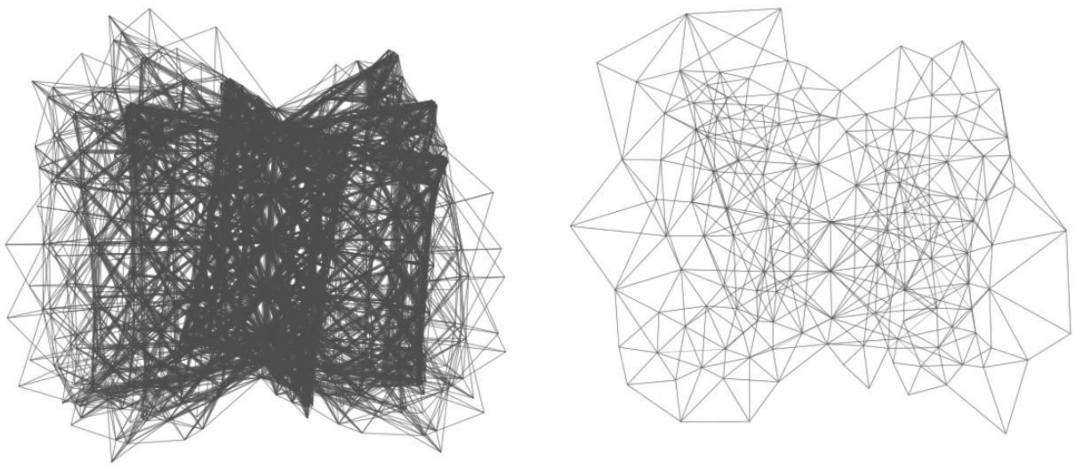
(a) original (n=16558, m=149658) (b) generalization (n=910, m=3520)

Figure 9.18: Generalization of the graph `PDS10` from the University of Florida sparse matrix collection [Dav94].



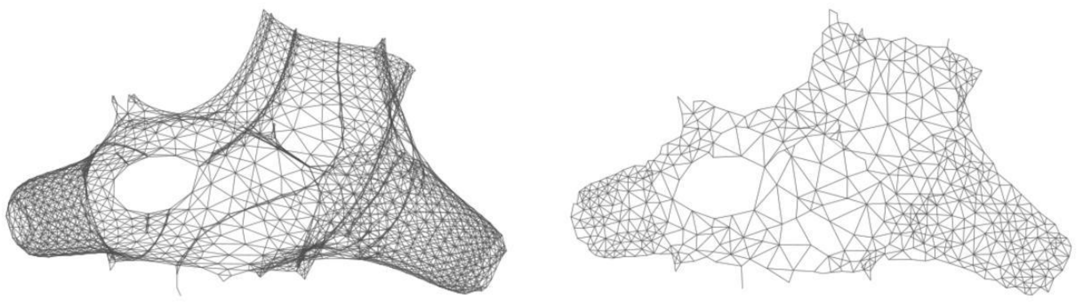
(a) original (n=1036, m=3736) (b) generalization (n=130, m=630)

Figure 9.19: Generalization of the graph `stufe` from the University of Florida sparse matrix collection [Dav94].



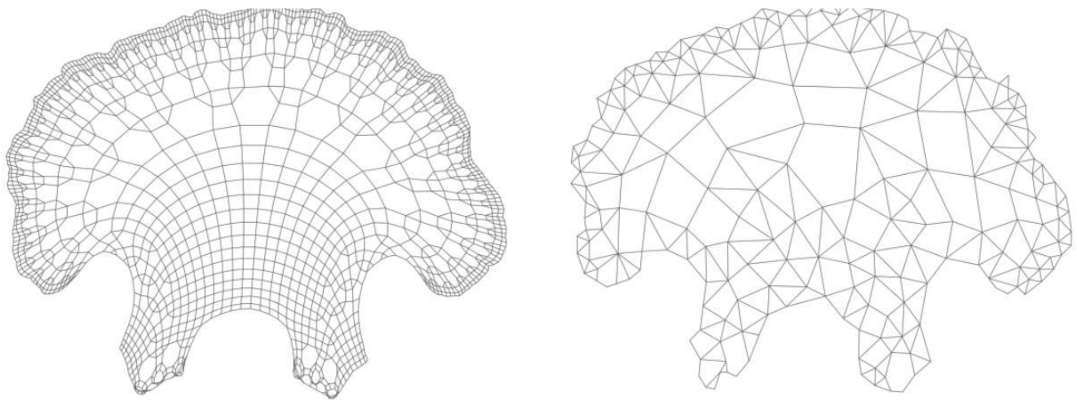
(a) original (n=1050, m=29156) (b) generalization (n=157, m=1110)

Figure 9.20: Generalization of the graph `msc01050` from the University of Florida sparse matrix collection [Dav94].



(a) original (n=1242, m=10426) (b) generalization (n=469, m=2608)

Figure 9.21: Generalization of the graph `dwt_1242` from the University of Florida sparse matrix collection [Dav94].



(a) original (n=1866, m=7076) (b) generalization (n=234, m=1062)

Figure 9.22: Generalization of the graph `ukerbe1_dual` from the University of Florida sparse matrix collection [Dav94].

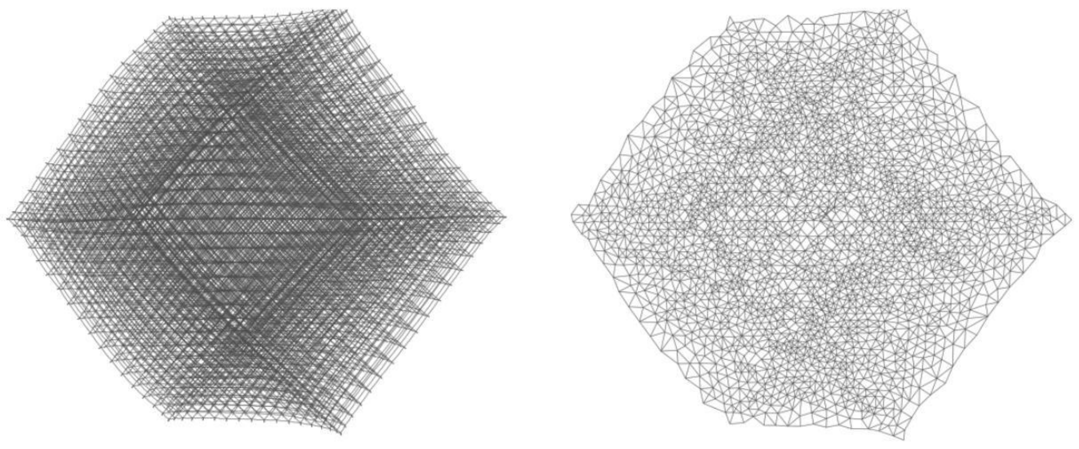

(a) original (n=24300, m=69984) (b) generalization (n=2093, m=13546)

Figure 9.23: Generalization of the graph `aug3d` from the University of Florida sparse matrix collection [Dav94].

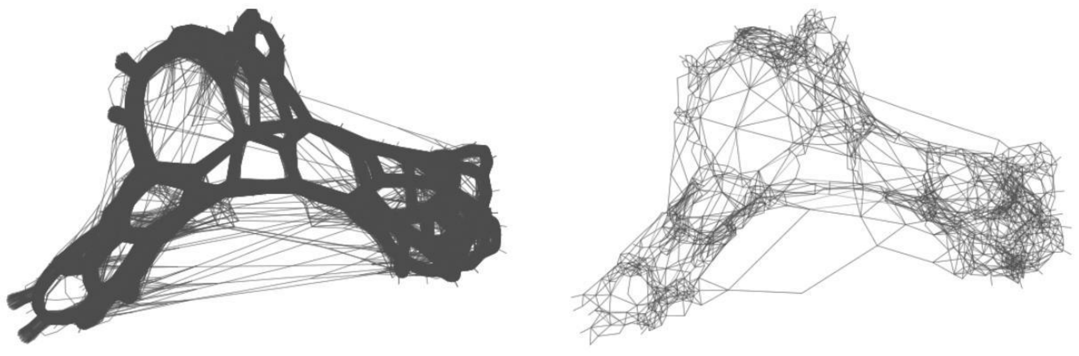

(a) original (n=44514, m=201050) (b) generalization (n=958, m=5763)

Figure 9.24: Generalization of the graph `lpl3` from the University of Florida sparse matrix collection [Dav94].

Table 9.1: Benchmark set of graphs used for our experiments, sorted according to number of vertices. All graphs, except those marked with ⋆, are from the University of Florida sparse matrix collection [Dav94]; `clique-planar` is a planar graph with an implanted clique, `lunar-vis` is a LunarVis layout of the Internet graph at the autonomous-systems level [GGW08], `email` is a force-based layout of the email network of the faculty of informatics at the Karlsruhe Institute of Technology and `osm_berlin` and `osm_isleofman` are street networks extracted from the OpenStreetMap data [osm11].

Index	**Name**	**Vertices**	**Edges**
1	bcspwr09	1036	3736
2	bcsstk08	1050	29156
3	bcsstk14	1072	12444
4	bcsstk26	1074	12960
5	can_1072	1133	10902
6	clique-planar⋆	1141	7465
7	bcsstk35	1242	10426
8	bcsstk36	1723	6511
9	bcsstk37	1733	22189
10	bodyy4	1806	63454
11	c-48	1821	52685
12	cti	1866	7076
13	ex3sta1	1919	32399
14	ford1	1922	30336
15	g7jac060sc	1960	11187
16	jan99jac060	1961	5156
17	jan99jac100sc	2363	7680
18	tandem_vtx	2423	11672
19	TF16	3345	19404
20	dwt_1242	6290	16466
21	email⋆	7920	31680
22	ex33	16558	149658
23	ex3	16782	678998
24	jagmesh8	16840	96464
25	aug3d	17233	74436
26	cep1	17546	121938
27	commanche_dual	18354	166080
28	conf5_4-8x8-05	18454	253350
29	lpl3	18728	101576
30	nemscem	23052	1143140
31	pds10	24300	69984
32	sstmodel	24494	51256
33	msc01050	25503	1140977
34	netz4504	30237	1450163
35	lunar-vis⋆	34758	432346
36	osm_berlin⋆	35460	406632
37	osm_isleofman⋆	41228	254364
38	plat1919	44514	201050
39	rajat02	49152	2064384
40	stufe	68908	431724
41	ukerbe1_dual	106675	248390

9.6 Conclusion and Open Problems

We have undertaken a first step at studying the problem of generalizing geometric graphs within a rigorous mathematical model. We formalized the problem by considering an incremental framework modeling the elimination or reduction of different types of clutter as optimization problems, which we analyzed in terms of complexity. Since these problems turned out to be NP-hard in general, we also devised efficient approximation algorithms as well as efficient heuristics. We showed how to heuristically eliminate vertex-clutter in $\mathcal{O}((n+m)\log^3 m + n\log^2 n)$ time and how to reduce edge clutter in $\mathcal{O}(nm+m^2)$ time considering geometric features such as point distributions and geodesic tendencies. After the elimination of vertex-clutter and edge-clutter we can expect the graph to be much smaller than the original graph. Hence, even larger complexities may scale accordingly. Thus, even the relatively high complexity of our heuristic for reducing vertex-edge clutter may be practical. Even without this step, however, the resulting generalizations exhibit considerably less clutter and are easier to analyze. We showcased some generalizations produced by our heuristics in Figures 9.14–9.24.

Open problems The problem of generalizing geometric graphs is problem with many facets. We have taken a first step at formalizing and studying the problem and designing efficient and practical algorithms. While the presented results are promising, there are many open problems. For instance, it is not clear if we can approximate both the local coverage and the size of a ϱ-set in the vertex generalization step. On the theoretical side, the complexity of the Local Coverage Cluster Packing problem remains un-answered for different type of mappings and the complexity of the optimal angle adjustment problem is also open. An interesting open problem with applications beyond the scope of the generalization problem is the question whether it is possible to approximate the size of a shortest geodesic subgraph, possibly in the presence of a limited drift. This problem is interesting in its own right since it relates to problems such as computing sparse spanners.

In addition to these problems we should study the determinants of a good generalization based on user studies. It is unclear how humans would go about the task of drawing a suitable generalization and we should try to find out which generalizations are considered good by users and which are not. Further, the generalization of geometric graphs should be considered in a dynamic scenario with user interaction. When zooming in an out, the various generalizations should be consistent.

Chapter 10

Conclusion

In this thesis we have studied a collection of various combinatorial and geometric network construction problems. First, we considered the problem of enumerating and generating degenerate graphs uniformly at random. We introduced a special new labeling scheme that is superior to the classical labeling scheme for degenerate graphs since the distribution of the unlabeled degenerate graphs induced by generating labeled degenerate graphs by this new labeling scheme is closer to the desired uniform distribution on the unlabeled graphs. We used this labeling scheme to design efficient uniform samplers and enumeration algorithms for the thus labeled degenerate graphs. Second, we turned our attention to classical augmentation problems on trees asking for an augmentation that is optimal with respect to routing cost and we presented algorithms for these problems that are optimal or near-optimal up to a logarithmic factor. Since realistic network construction problems often need to find a balance between several optimization criteria we studied a framework of network construction problems with one maximization goal and one minimization goal, which we combined by considering the ratio of the two goals. We performed a thorough study of the complexity of this problem for various classes of graphs and under various constraints and we presented hardness and inapproximability results as well as efficient algorithms for restricted or relaxed variants of the problem, including a fully polynomial-time approximation scheme. Interestingly, a recurring pattern was that we could solve some of the purely combinatorial problems using concepts from geometry, even though the considered problem variants did not have an immediate connection to geometry. While, in some sense, a similar argument could be made for all problems that can be formulated within the framework of linear programming, we made a more direct use of geometry by using such concepts as upper envelopes, tangent query and Voronoi diagrams, that is, classical problems from computational geometry.

In the second part of this thesis, we considered geometric network construction problems. First, we studied various network embedding problems. We introduced a new drawing style based on shortest orthogonal links and studied both algorithmic and combinatorial properties of various variants of the point-set embedding problems for this drawing style. We presented several hardness-results and efficient algorithms. Further, we studied the orthogeodesic embedding problem from a combinatorial perspective and provided upper bounds for the minimum value $f(n)$ such that all general point sets with $f(n)$ points are universal for various classes of trees with n vertices subject to various restrictions on the embedding. Additionally, we studied a colored version of the orthogeodesic embedding problem for paths and matchings and provided NP-hardness results and an approximation algorithms on the grid as well as an efficient exact algorithm without the restriction to the grid. Finally, in Chapter 9, we studied the problem of generalization geometric graphs, that is, the problem of constructing a small network that both visually and structurally resembles a given larger network. This problem has applications in network visualization and cartography. We presented a first formalization of this problem to its full extent in a formal mathematical model and studied the complexity of the resulting problems. While the resulting optimization problems are NP-hard, we further presented efficient approximation algorithms as well as effective and easy-to-implement heuristics and showcased the resulting generalizations.

Outlook

We have already pointed out various directions for future research concerning the specific problems considered in this thesis in the respective chapters. Therefore, we address some of the computational network construction problems from a broader point of view here.

Artificially Generated Network Data As networks are rapidly developing in growth and as they are increasingly finding their way into various applications and everyday life—from social networks and their analysis over route planning applications to complex integrated circuit design—the need for algorithmic solutions to the various kinds of computational problems involving networks will probably increase and the interest in algorithms for networks will gain further momentum. Efficient algorithms for these tasks are often obtained by combining theoretical algorithmic results with experience from practical implementations. Algorithm engineering is therefore built on not only on the theory of algorithms but also on the experimental analysis of implementations.

While highly efficient algorithms are typically custom-tailored for a specific application and the respective algorithms are therefore usually tested on real-word data, there are various reasons for relying not only on real-world data but also on artificially generated data when performing an experimental study for an implementation. On the one hand, real-world data may be hard to obtain, for instance, if the data is sensitive or if it is not yet available. On the other hand, even if plenty of data is available, there are good reasons for additionally using artificially generated data for experiments. For instance, it may be expected that future networks for the application that is being developed will be much larger than the networks that are currently available. In this case, large, artificially generated networks can help assessing whether algorithms are *scalable*. Another reason is that performing experiments solely on real-world data will, in some sense, result in overfitting the algorithm to the test data. If the underlying networks change over time, these algorithms may not be as efficient on the modified networks as they have been on the original networks on which they have been tested. In this case, testing with artificially generated data will increase the *robustness* of algorithms in terms of efficiency. Finally, testing algorithms with artificially generated data is more likely to reveal errors in the implementation since the characteristics of uniformly generated artificial network data are typically less biased than those of real-world data, and therefore, artificially generated data is more likely to exhibit characteristics that are not present in real-world data.

Since realistic networks often exhibit specific graph-theoretic properties, the need for custom-tailored graph generators capable of generating network data with specified properties will probably increase as the number of algorithmic solutions for network tasks increases.

Generating graphs with non-trivial properties involves many interesting and complex combinatorial and algorithmic problems and constitutes an interesting field of future research.

Apart from studying generators for these problems from a combinatorial point of view, however, a major challenge is to reduce the gap between theory and practice. While there are major research efforts on the theory of uniform graph generators for various classes of graph, algorithms for generating graphs are in practice often implemented on the fly to satisfy the experimenters' current needs. Since the choice of the network generator may, however, have a severe effect on the outcome of systematic tests on the data, it is important to use uniform and, where this is not possible, at least complete generators. However, only efficient and easy-to-implement uniform algorithms will eventually find their way into widely used test suites.

Interactive Network Design As network construction tasks are becoming increasingly more complex and time-consuming, many network construction problems that are still solved manually today will have to be solved with the aid of computers in an at least partially automated way in the future. However, the increasing complexity of these problems also makes it harder to formalize the network construction problems within a rigorous mathematical model and, even if this succeeds, the resulting formulations are often complicated to tackle algorithmically. As an example, consider the problem of generalizing geometric graphs that we studied in Chapter 9. While we do have a fairly good intuitive understanding this task it is nevertheless far from being straight-forward to formalize. This is mainly due to the complex set of rules governing the generalization process with respect to geometry, topology and visual appearance as well as multiple, conflicting optimization goals.

On the other hand, many realistic network construction problems involve domain-specific constraints and optimization goals and, therefore, algorithms for these problems need to be adaptable in an easy way. Generalizing a geographic map, for instance, is different from generalizing a visualization of a social network, although these problems do have many features in common. Additionally, it is to be expected that algorithms for network construction problems will increasingly be used by non-experts who will nevertheless wish to adapt algorithms in order to compute custom-tailored solutions. As an example consider a route planning system for mobile navigation in cars. While it is reasonable to assume that users would like to compute the shortest route with respect to some metric, many users will have different preferences and needs and will, therefore, prefer slightly different routes.

Algorithm designers must therefore devise algorithms with intuitive and adaptive interfaces that are capable of computing good solutions to complex network construction tasks in which the optimum is not clearly defined, either due to a lack of a rigorous mathematical model or in the case when there are several conflicting optimization goals. While most people using mobile navigation gadgets know that it can be quite annoying if the suggested route does not meet his or her expectations, there is little that users can do against it at the moment. A first approach to this problem is to present users with several Pareto-optimal alternatives, thus, providing the user with the possibility to interact with the algorithm. However, this does not solve the problem to a satisfactory extent.

With more complex network construction tasks, such as the generalization of geometric graphs or transportation problems evolving in logistics, the set of reasonable alternatives may be too large and no set of alternative solutions may be suitable for display to a user.

Therefore, we need more elaborate mechanisms for interaction with the algorithms. Given an initial solution that is optimal with respect to some set of constraints and optimization goals we may subsequently wish to refine this solution in multiple ways. For instance, we may wish to avoid only a part of a computed route or we would like to reduce the amount of displayed data in only a certain part of the computed generalization while increasing the amount in other parts of the generalization. Typically, however, it is not desired to compute a new solution from scratch, but to modify a given solution in the least possible so as to incorporate the new restrictions.

Although many network augmentation algorithms can be used in an interactive scenario already and although interaction is, in principle, supported by generic techniques such as constraint programming and integer programming, the implications of interaction are only just beginning to be studied from an algorithmic point of view and may therefore constitute an interesting field of future research with many applications.

Geometric Network Construction and Network Embedding Network embedding plays a role in both the construction of tangible networks and in the visualization of network-based data. With the increasing use of computer-aided design not only for integrated circuits but also, for instance, in architectural applications, we are facing many challenging computational network construction problems. For instance, we may wish to automatically compute the layout of the electrical wiring in a large housing complex. The design of such a network is underlying a complex set of constraints resulting from the geometry of the building and technical building regulations. On the other hand, such a network should be failsafe, serviceable and not too expensive. These restrictions affect both the geometrical features of the network and the structural properties alike. To the best of our knowledge, this problem has not yet been thoroughly addressed and there are many similar geometric network construction problems that may yield interesting directions for future research.

Further, as network-based data is increasing and since visualization of data in the network-metaphor seems to be accessible to users in an intuitive way, we are likely to witness an increasing interest in graph drawing as well as new fields of application for network visualization. But even today, network visualization already plays an important role for software-engineers, biologists, sociologists and many more. While there is a multitude of generic algorithms for visualizing network data, there are many challenging tasks with respect to integrating a multitude of additional domain-specific constraints. Apart from handling these constraints, network visualization tools must provide intuitive mechanisms for manipulating the resulting visualizations in order to readjust parts of the automatic layout the user does not like. This, too, is an interaction of the user with the layout algorithm and presents new interesting problems in graph drawing that are only just beginning to be studied. Further, we are still a rather long way from understanding the aesthetic rules guiding humans in the task of network visualization. Only this understanding, however, can enable algorithm designers to further enhance automatically computed layouts. Therefore, we will need to devise algorithms that can be custom-tailored in multiple ways and we need to put more effort into understanding the users' preferences.

Additionally, since network visualization is often used for network analysis, we need to generate analytical layouts that are capable of conveying meaningful analytics and significant statistics of the underlying data to enable decision making based on these layouts. A key challenge in this field of application is to find layouts that are both aesthetically pleasing and convey reliable information on the data.

Finally, with the increasing use of network visualization we need adaptive and versatile generalization techniques to display the visualizations on a multitude of different display media. This is even further emphasized by the increasing use of office applications on small screens of mobile devices. While generalization used to be done by hand, for instance, in cartography for a long time, the increasing availability of network data and the fact that layouts are increasingly computed on the fly, highlights the need for automatic generalization. As in the visualization of network-based data, we are only just beginning to understand the mechanisms underlying this complex task and there are many related open problems for future research.

Although network construction problems with various characteristics have been studied extensively, there are still many interesting open problems and future challenges. Therefore, computational network construction will remain an interesting and important area of research in computer science.

Bibliography

[ABCC06] David L. Applegate, Robert E. Bixby, Vašek Chvátal, and William J. Cook. *The Traveling Salesman Problem: A Computational Study.* Princeton University Press, 2006. 3

[ACD$^+$11] Patrizio Angelini, Enrico Colasante, Giuseppe Di Battista, Fabrizio Frati, and Maurizio Patrignani. Monotone drawings of graphs. In Ulrik Brandes and Sabine Cornelson, editors, *Proc. 18th Internat. Symp. Graph Drawing (GD'10)*, volume 6502 of *Lecture Notes Comput. Sci.*, pages 13–24. Springer-Verlag, 2011. 120

[ACG$^+$02] Giorgio Ausiello, Pierluigi Crescenzi, Giorgio Gambosi, Viggo Kann, and Alberto Marchetti-Spaccamela. *Complexity and Approximation – Combinatorial Optimization Problems and Their Approximability Properties.* Springer-Verlag, 2nd edition, 2002. 18

[AFG09] Patrizio Angelini, Fabrizio Frati, and Luca Grilli. An algorithm to construct greedy drawings of triangulations. In Ioannis G. Tollis and Maurizio Patrignani, editors, *Proc. 8th Internat. Symp. Graph Drawing (GD'08)*, volume 5417 of *Lecture Notes Comput. Sci.*, pages 26–37. Springer-Verlag, 2009. 120

[AG09] Noga Alon and Shai Gutner. Linear time algorithms for finding a dominating set of fixed size in degenerated graphs. *Algorithmica*, 54:544–556, 2009. 10.1007/s00453-008-9204-0. 22

[AGHT03] Manuel Abellanas, Alfredo García, Ferran Hurtado, and Javier Tejel. Caminos alternantes (in Spanish). In *X Encuentros de Geometría Computacional, Sevilla, Spanish 2003*, pages 7–12, 2003. (English version available on Ferran Hurtado's web page). 170

[AGLHP$^+$99] Manuel Abellanas, Jesus Garcia-Lopez, Gregorio Hernández-Peñalver, Marc Noy, and Pedro A. Ramos. Bipartite embeddings of trees in the plane. *Discrete Applied Mathematics*, 93(2-3):141–148, 1999. 170

[AKF01] James Abello, Jeffrey Korn, and Irene Finocchi. Graph sketches. In *Proc. IEEE Symp. Information Visualization 2001 (INFOVIS'01)*, pages 67–71. IEEE Computer Society, 2001. 198

[AKY05] James Abello, Stephen Kobourov, and Roman Yusufov. Visualizing large graphs with compound-fisheye views and treemaps. In János Pach, editor, *Graph Drawing*, volume 3383 of *Lect. Not. Comput. Sci.*, pages 431–441. Springer-Verlag, Berlin / Heidelberg, 2005. 198

[And79] A. M. Andrew. Another efficient algorithm for convex hulls in two dimensions. *Inform. Process. Lett.*, 9(5):216–219, 1979. 80

[AS72] Milton Abramowitz and Irene A. Stegun, editors. *Handbook of Mathematical Functions With Formulas, Graphs, and Mathematical Tables*. Dover, 10 edition, 1972. 38

[AU90] Jin Akiyama and Jorge Urrutia. Simple alternating path problem. *Discrete Mathematics*, 84(1):101–103, 1990. 170

[AYZ95] Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995. 114

[Bál03] Vojtech Bálint. The non-approximability of bicriteria network design problems. *J. of Discrete Algorithms*, 1:339–355, June 2003. 91

[BDG$^+$08] Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Görke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. On modularity clustering. *IEEE Trans. Knowledge and Data Engineering*, 20:172–188, 2008. 201

[BDL08] Melanie Badent, Emilio Di Giacomo, and Giuseppe Liotta. Drawing colored graphs on colored points. *Theoretical Computer Science*, 408(2-3):129–142, 2008. 148

[Ber41] Andrew C. Berry. The accuracy of the gaussian approximation to the sum of independent variates. *Transactions of the American Mathematical Society*, 49(1):122–136, 1941. 31, 32

[BETT99] Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis, editors. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1999. 3, 14

[BFW73] D. E. Boyce, A. Farhi, and R. Weischedel. Optimal network problem: a branch-and-bound algorithm. *Environment and Planning*, 5(4):519–533, 1973. 3

[BGG$^+$07] Michael Baur, Marco Gaertler, Robert Görke, Marcus Krug, and Dorothea Wagner. Generating Graphs with Predefined k-Core Structure. In *Proceedings of the European Conference of Complex Systems (ECCS'07)*, October 2007. 24

[BGG$^+$08] Michael Baur, Marco Gaertler, Robert Görke, Marcus Krug, and Dorothea Wagner. Augmenting k-Core Generation with Preferential Attachment. *Networks and Heterogeneous Media*, 3(2):277–294, June 2008. 22

[BGK07] Manuel Bodirsky, Clemens Gröpl, and Mihyun Kang. Generating labeled planar graphs uniformly at random. *Theor. Comput. Sci.*, 379(3):377–386, 2007. 23

[BGK$^+$12] Edith Brunel, Andreas Gemsa, Marcus Krug, Ignaz Rutter, and Dorothea Wagner. Generalizing Geometric Graphs. In *Proceedings of the 19th International Symposium on Graph Drawing (GD'11)*, Lecture Notes in Computer Science. Springer-Verlag, 2012. to appear. 197

[BK94] Therese Biedl and Goos Kant. A better heuristic for orthogonal graph drawings. In Jan van Leeuwen, editor, *Proc. 2nd Ann. Europ. Symp. Algorithms (ESA'94)*, volume 855 of *Lecture Notes Comput. Sci.*, pages 24–35. Springer-Verlag, 1994. 123

[BK08] Hans L. Bodlaender and Arie M. Koster. Combinatorial optimization on graphs of bounded treewidth. *The Computer Journal*, 51(3):255–269, May 2008. 11

[BKN08] Béla Bollobás, Alexandr Kostochka, and Kittikorn Nakprasit. Packing d-degenerate graphs. *Journal of Combinatorial Theory, Series B*, 98(1):85–94, 2008. 22

[BKRW10] Thomas Bläsius, Marcus Krug, Ignaz Rutter, and Dorothea Wagner. Drawing orthogonal graphs with flexibility constraints. In Ulrik Brandes and Sabine Cornelsen, editors, *Proc. 18th Internat. Symp. Graph Drawing (GD'10)*, volume 6502 of *Lecture Notes Comput. Sci.*, pages 92–104. Springer-Verlag, 2010. 122, 123

[BKW10] Reinhard Bauer, Marcus Krug, and Dorothea Wagner. Enumerating and Generating Labeled k-Degenerate Graphs. In *Proceedings of the 7th Workshop on Analytic Algorithmics and Combinatorics (ANALCO '10)*, pages 90–98. SIAM, 2010. 19

[BM07] Gunnar Brinkmann and Brendan D. McKay. Fast generation of planar graphs. *MATCH - Communications in Mathematical and in Computer Chemistry*, 58(2):323–357, 2007. 23

[BMS97] Prosenjit Bose, Michael McAllister, and Jack Snoeyink. Optimal algorithms to embed trees in a point set. *Journal of Graph Algorithms and Applications*, 2(1):1–15, 1997. 147

[Bod93] Hans L. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. In *STOC '93: Proc. 25th Ann. ACM Symp. Theory of Computing*, pages 226–234, New York, NY, USA, 1993. ACM. 99

[boo] Boost C++ libraries, version 1.42. http://www.boost.org. 216

[Bor26] Otakar Borůvka. O Jistém Problému Minimálním (About a Certain Minimal Problem) (in Czech, German summary). *Práce Mor. Prírodoved. Spol. v Brne III*, 3, 1926. 2

[Bos02] Prosenjit Bose. On embedding an outer-planar graph on a point set. *Computational Geometry: Theory and Applications*, 23:303–312, 2002. 147

[Bra08] Franz J. Brandenburg. Drawing planar graphs on $\frac{8}{9}n^2$ area. *Electronic Notes in Discrete Mathematics*, 31:37–40, 2008. 148

[BS80] Jon Louis Bentley and James B. Saxe. Decomposable searching problems I. Static-to-dynamic transformation. *Journal of Algorithms*, 1(4):301–358, 1980. 208

[BS09] Roger E. Bohn and James E. Short. How much information? 2009 Report on American consumers. Global Information Industry Center, University of California, San Diego, 2009. 198

[BZ11] Vladimir Batagelj and Matjaž Zaveršnik. Fast algorithms for determining (generalized) core groups in social networks. *Advances in Data Analysis and Classification*, 5:129–145, 2011. 10.1007/s11634-010-0079-y. 22

[Cab06] Sergio Cabello. Planar embeddability of the vertices of a graph using a fixed point set is NP-hard. *J. Graph Algorithms Appl.*, 10(2):353–363, 2006. 120, 147

[Car08] Juan-Antonio Carballo. *Chip Design for Non-Designers: An Introduction.* PennWell Corp., 2008. 2

[CCC06] Leizhen Cai, Siu Man Chan, and Siu On Chan. Random separation: A new method for solving fixed-cardinality optimization problems. In *Proc. 2nd Internat. Workshop on Parameterized and Exact Computation (IWPEC'06)*, pages 239–250, 2006. 22

[CCJ90] Brent N. Clark, Charles J. Colbourn, and David S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86(1-3):165–177, 1990. 202

[CCPY86] Bernard Chazelle, Richard Cole, Franco P. Preparata, and Chee-Keng Yap. New upper bounds for neighbor searching. *Information and Control*, 68(1-3):105–124, 1986. 207

[cga] CGAL, Computational Geometry Algorithms Library. http://www.cgal.org. 216

[CGS11] Francis Chin, Zuyu Guo, and He Sun. Minimum Manhattan network is NP-complete. *Discrete Comput. Geom.*, 45:701–722, 2011. 121

[Cha77] R. Chandrasekaran. Minimal ratio spanning trees. *Networks*, 7(4):335–342, 1977. 92, 107

[Cha88] Bernard Chazelle. Functional approach to data structures and its use in multidimensional searching. *SIAM J. Comput.*, 17:427–462, June 1988. 176, 208

[CKM$^+$09] Josef Cibulka, Jan Kynčl, Viola Mészáros, Rudolf Stolař, and Pavel Valtr. Hamiltonian alternating paths on bicolored double-chains. In Ioannis G. Tollis and Maurizio Patrignani, editors, *Graph Drawing*, volume 5417 of *Lecture Notes Comput. Sci.*, pages 181–192. Springer-Verlag, 2009. 170

[CL05] Kai-min Chung and Hsueh-I Lu. An optimal algorithm for the maximum-density segment problem. *SIAM J. Comput.*, 34(2):373–387, 2005. 92

[CLRS09] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms.* Mit Press, 3 edition, 2009. 9, 15

[CP07] Altannar Chinchuluun and Panos Pardalos. A survey of recent developments in multiobjective optimization. *Annals of Operations Research*, 154:29–50, 2007. 90, 91

[Cre97] Pierluigi Crescenzi. A short guide to approximation preserving reductions. In *Proc. 12th Ann. IEEE Conf. Computational Complexity*, pages 262–273, Washington, DC, USA, 1997. IEEE Computer Society. 18

[CZ01] Leizhen Cai and Xuding Zhu. Game chromatic index of k-degenerate graphs. *J. Graph Theory*, 36(3):144–155, 2001. 22

[Dav94] Timothy A. Davis. University of florida sparse matrix collection. *NA Digest*, 92, 1994. 216, 219, 222, 223, 224, 225, 226

[dBK10] Mark de Berg and Amirali Khosravi. Optimal binary space partitions in the plane. In M. Thai and S. Sahni, editors, *Computing and Combinatorics*, volume 6196 of *Lect. Not. Comput. Sci.*, pages 216–225. Springer-Verlag, Berlin / Heidelberg, 2010. 202

[DBT88] Giuseppe Di Battista and Roberto Tamassia. Algorithms for plane representations of acyclic digraphs. *Theor. Comput. Sci.*, 61:175–198, 1988. 120

[Dem07] Erik Demaine. Simple polygonizations. http://erikdemaine.org/polygonization/, 2007. Accessed May 30, 2009. 121

[Dev86] Luc Devroye. *Non-Uniform Random Variate Generation.* Springer-Verlag, Berlin / Heidelberg, 1986. 25, 28, 30

[DF79] René Dionne and Michael Florian. Exact and approximate algorithms for optimal network design. *Networks*, 9:37–60, 1979. 3, 68

[DF95] Rod G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness II: On completeness for W[1]. *Theoretical Computer Science*, 141(1-2):109–131, 1995. 113

[DFS90] David Dobkin, Steven Friedman, and Kenneth Supowit. Delaunay graphs are almost as good as complete graphs. *Discrete & Computational Geometry*, 5:399–407, 1990. 213

[DGFF$^+$12] Emilio Di Giacomo, Fabrizio Frati, Radoslav Fulek, Luca Grilli, and Marcus Krug. Orthogeodesic Point-Set Embedding of Trees. In *Proceedings of the 19th International Symposium on Graph Drawing (GD'11)*, Lecture Notes in Computer Science. Springer-Verlag, 2012. to appear. 147

[DGGK$^+$12] Emilio Di Giacomo, Luca Grilli, Marcus Krug, Giuseppe Liotta, and Ignaz Rutter. Hamiltonian Orthogeodesic Alternating Paths. In *Proceedings of the 22nd International Workshop on Combinatorial Algorithms*, Lecture Notes in Computer Science. Springer-Verlag, 2012. to appear. 169

[DKS09] Guoli Ding, Jinko Kanno, and Jianning Su. Generating 5-regular planar graphs. *Journal of Graph Theory*, 61(3):219–240, July 2009. 24

[DLT10] Emilio Di Giacomo, Giuseppe Liotta, and Francesco Trotta. Drawing colored graphs with constrained vertex positions and few bends per edge. *Algorithmica*, 57:796–818, 2010. 148

[DS98] Egbert Dierker and Karl Sigmund, editors. *Karl Menger: Ergebnisse eines mathematischen Kolloquiums*, chapter Bericht über das Kolloquium 1929/130., page 130. Springer-Verlag, Wien/New York, 1998. 2

[DVW96] Alain Denise, Marcio Vasconcellos, and Dominic J. A. Welsh. The random planar graph. *Congressus Numerantium*, 113:61–79, 1996. 23

[DW71] S. Dreyfus and R. Wagner. The Steiner problem in graphs. *Networks*, 1(3):195–207, 1971. 92

[DWW$^+$11] Ling Ding, Weili Wu, James K. Willson, Hongjie Du, and Wonjun Lee. Construction of directional virtual backbones with minimum routing cost in wireless networks. In *INFOCOM, 2011 Proceedings IEEE*, pages 1557–1565, April 2011. 3

[EF97] Peter Eades and Qing-Wen Feng. Multilevel visualization of clustered graphs. In Stephen North, editor, *Graph Drawing*, volume 1190 of *Lecture Notes Comput. Sci.*, pages 101–112. Springer-Verlag, Berlin / Heidelberg, 1997. 198

[EGK$^+$03] John Ellson, Emden R. Gansner, Eleftherios Koutsofios, Stephen C. North, and Gordon Woodhull. Graphviz and dynagraph – static and dynamic graph drawing tools. In M. Junger and P. Mutzel, editors, *Graph Drawing Software*, pages 127–148. Springer-Verlag, 2003. 216

[EGS89] Herbert Edelsbrunner, Leonidas Guibas, and Micha Sharir. The upper envelope of piecewise linear functions: Algorithms and applications. *Discrete & Computational Geometry*, 4:311–336, 1989. 215

[EH66] Paul Erdős and András Hajnal. On chromatic number of graphs and set-systems. *Acta Mathematica Hungarica*, 17:61–99, 1966. 10.1007/BF02020444. 22

[ELLW10] Hazel Everett, Sylvain Lazard, Giuseppe Liotta, and Stephen K. Wismath. Universal sets of n points for one-bend drawings of planar graphs with n vertices. *Discrete Comput. Geom.*, 43:272–288, 2010. 120, 148

[Epp95] David Eppstein. Subgraph isomorphism in planar graphs and related problems. In *Proc. 6th Ann. ACM-SIAM Symp. Discrete Algorithms*, pages 632–640. SIAM, 1995. 99

[Erw00] Martin Erwig. The graph Voronoi diagram with applications. *Networks*, 36(3):156–163, 2000. 74

[ES35] Paul Erdős and George Szekeres. A combinatorial problem in geometry. *Compositio Mathematica*, 2:463–470, 1935. 158, 159

[ES05] Niklas Eén and Niklas Sörensson. MiniSat v1. 13-a SAT solver with conflict-clause minimization. *SAT*, 2005:2–3, 2005. 166

[Ess42] Carl-Gustav Esseen. On the Liapunoff limit of error in the theory of probability. *Arkiv for Matematik, Astronomi och fysik*, A28(9):1–19, 1942. 31, 32

[Ess56] Carl-Gustav Esseen. A moment inequality with an application to the central limit theorem. *Skandinavisk Aktuarietidskrift*, 39:160–170, 1956. 31, 32

[FG06] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory.* Springer-Verlag, 2006. 17

[FGG08] Mohammad Farshi, Panos Giannopoulos, and Joachim Gudmundsson. Improving the stretch factor of a geometric network by edge augmentation. *SIAM J. Comput.*, 38:226–240, March 2008. 75

[FLS02] Matteo Fischetti, Giuseppe Lancia, and Paolo Serafini. Exact algorithms for minimum routing cost trees. *Networks*, 39(3):161–173, 2002. 68

[For87] Steven Fortune. A sweepline algorithm for Voronoi diagrams. *Algorithmica*, 2:153–174, 1987. 72, 73

[Fra94] András Frank. *Mathematical Programming: State of the Art*, chapter Connectivity augmentation problems in network design, pages 34–63. University of Michigan, 1994. 68

[FS09] Philippe Flajolet and Robert Sedgewick. *Analytic Combinatorics.* Cambridge University Press, Cambridge, January 2009. 20

[Fur86] George W. Furnas. Generalized fisheye views. *SIGCHI Bull.*, 17:16–23, April 1986. 198

[Fus09] Éric Fusy. Uniform random sampling of planar graphs in linear time. *Randoms Structures and Algorithms*, 35(4):464–522, December 2009. 23

[Gab83] Harold N. Gabow. An efficient reduction technique for degree-constrained subgraph and bidirected network flow problems. In *STOC'83: Proc. 15th Ann. ACM Symp. Theory of computing*, pages 448–456, New York, NY, USA, 1983. ACM. 63

[Gab90] Harold N. Gabow. Data structures for weighted matching and nearest common ancestors with linking. In *Proc. 1st Ann. ACM-SIAM Symp. Discrete Algorithms*, SODA '90, pages 434–443, Philadelphia, PA, USA, 1990. Society for Industrial and Applied Mathematics. 110

[Gae05] Marco Gaertler. Clustering. In U. Brandes and T. Erlebach, editors, *Network Analysis*, volume 3418 of *Lect. Not. Comput. Sci.*, pages 178–215. Springer-Verlag, Berlin / Heidelberg, 2005. 198, 201

[GGW08] Robert Görke, Marco Gaertler, and Dorothea Wagner. Lunarvis - analytic visualizations of large graphs. In *Proc. 15th Internat. Conf. Graph Drawing*, GD'07, pages 352–364, Berlin / Heidelberg, 2008. Springer-Verlag. 216, 222, 226

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, 1979. 15, 16, 92, 112, 124, 134, 182, 209

[GK11] Sachin Garg and Gaurav Kanade. Topology construction for rural wireless mesh networks – a geometric approach. In Beniamino Murgante, Osvaldo Gervasi, Andrés Iglesias, David Taniar, and Bernady Apduhan, editors, *Computational Science and Its Applications - ICCSA 2011*, volume 6784 of *Lecture Notes Comput. Sci.*, pages 107–120. Springer-Verlag, Berlin / Heidelberg, 2011. 10.1007/978-3-642-21931-3_9. 3

[GKL05] Michael H. Goldwasser, Ming-Yang Kao, and Hsueh-I Lu. Linear-time algorithms for computing maximum-density sequence segments with bioinformatics applications. *J. Comput. Syst. Sci.*, 70(2):128–144, 2005. 92

[GKN05] Emden R. Gansner, Yehuda Koren, and Stephen C. North. Topological fisheye views for visualizing large graphs. *Visualization and Computer Graphics, IEEE Transactions on*, 11(4):457–468, July-August 2005. 198

[GKO$^+$09] Xavier Goaoc, Jan Kratochvíl, Yoshio Okamoto, Chan-Su Shin, Andreas Spillner, and Alexander Wolff. Untangling a planar graph. *Discrete Comput. Geom.*, 42(4):542–569, 2009. 121

[GMPP91] Peter Gritzmann, Bojan Mohar, János Pach, and Richard Pollack. Embedding a planar triangulation with vertices at specified points. *Amer. Math. Monthly*, 98(2):165–166, 1991. 147

[GO97] Jacob E. Goodman and Joseph O'Rourke, editors. *Handbook of Combinatorial and Computational Geometry,*, chapter Pseudoline Arrangements. CRC Press, 1997. 78

[Gro92] Harald Gropp. Enumeration of regular graphs 100 years ago. *Discrete Mathematics*, 101(1-3):73–85, 1992. 24

[GT01] Ashim Garg and Roberto Tamassia. On the computational complexity of upward and rectilinear planarity testing. *SIAM J. Comput.*, 31(2):601–625, 2001. 123

[GV08] Petr Golovach and Yngve Villanger. Parameterized complexity for domination problems on degenerate graphs. In Hajo Broersma, Thomas Erlebach, Tom Friedetzky, and Daniel Paulusma, editors, *Graph-Theoretic Concepts in Computer Science*, volume 5344 of *Lecture Notes Comput. Sci.*, pages 195–205. Springer-Verlag, Berlin / Heidelberg, 2008. 10.1007/978-3-540-92248-3_18. 22

[HC05] Sun-Yuan Hsieh and Ting-Yu Chou. *Algorithms and Computation*, volume 3827 of *Lecture Notes Comput. Sci.*, chapter Finding a Weight-Constrained Maximum-Density Subtree in a Tree, pages 944–953. Springer-Verlag, Berlin / Heidelberg, 2005. 92

[HC08] Sun-Yuan Hsieh and Chih-Sheng Cheng. Finding a maximum-density path in a tree under the weight and length constraints. *Information Processing Letters*, 105(5):202–205, 2008. 92

[HEH09] Weidong Huang, Peter Eades, and Seok-Hee Hong. A graph reading behavior: Geodesic-path tendency. In *Proc. IEEE Pacific Symp. Visualization (PacificVis'09)*, pages 137–144, 2009. 120, 208

[HJ05] Stefan Hachul and Michael Jünger. Drawing large graphs with a potential-field-based multilevel algorithm. In János Pach, editor, *Graph Drawing*, volume 3383 of *Lect. Not. Comput. Sci.*, pages 285–295. Springer-Verlag, Berlin / Heidelberg, 2005. viii, 197

[HK02a] David Harel and Yehuda Koren. A fast multi-scale method for drawing large graphs. *Journal of graph algorithms and applications*, 6:179–202, 2002. 198

[HK02b] David Harel and Yehuda Koren. Graph drawing by high-dimensional embedding. In *Graph Drawing (GD'02)*, Lect. Not. Comput. Sci., pages 207–219. Springer-Verlag, 2002. 197

[HKLS04] Ferran Hurtado, Rolf Klein, Elmar Langetepe, and Vera Sacristán. The weighted farthest color Voronoi diagram on trees and graphs. *Computational Geometry*, 27(1):13–26, 2004. 74

[How10] Tharon Howard. *Design to Thrive: Creating Social Networks and Online Communities that Last.* Morgan Kaufmann, 2010. 2

[HR97] Magnus Halldórsson and Jaikumar Radhakrishnan. Greed is good: Approximating independent sets in sparse and bounded-degree graphs. *Algorithmica*, 18:145–163, 1997. 207

[Hur06] Ferran Hurtado. Personal communication, 2006. 121, 122

[HvW09] Danny Holten and Jarke J. van Wijk. Force-directed edge bundling for graph visualization. In *Proc. 11th Eurographics/IEEE-VGTC Symp. Visualization*, pages 983–990, 2009. 198

[IAN10] Toshimasa Ishii, Yoko Akiyama, and Hiroshi Nagamochi. Minimum augmentation of edge-connectivity between vertices and sets of vertices in undirected graphs. *Algorithmica*, 56(4):413–436, 2010. 68

[Inc08] Wolfram Research, Inc. Mathematica, version 7.0. Champaign, Illinois, 2008. 38

[Inm66] Ross B. Inman. A denaturation map of the lambda phage DNA molecule determined by electron microscopy. *Journal of Molecular Biology*, 18(3):464–476, 1966. 92

[JLK78] David S. Johnson, Jan K. Lenstra, and Alexander H. G. Rinnooy Kan. The complexity of the network design problem. *Networks*, 8(4):279–285, 1978. 3, 68

[JSJK06] Manoj K. Jha, Paul Schonfeld, J.-C. Jong, and E. Kim. *Intelligent Road Design*. WIT Press, 2006. 2

[Kan09] Mikio Kano. Discrete geometry on red and blue points on the plane lattice. In *Proc. Japan Conf. Computational Geometry and Graphs (JCCGG 2009)*, pages 30–33, 2009. 169, 171, 180, 185, 195

[KCH03] Yehuda Koren, Liran Carmel, and David Harel. Drawing huge graphs by algebraic multigrid optimization. *Multiscale Modeling and Simulation*, 1:645–673, 2003. viii, 197

[Kir83] David Kirkpatrick. Optimal search in planar subdivisions. *SIAM Journal on Computing*, 12(1):28–35, 1983. 72

[KKK$^+$11a] Mong-Jen Kao, Bastian Katz, Marcus Krug, D.T. Lee, Martin Nöllenburg, Ignaz Rutter, and Dorothea Wagner. Connecting Two Trees with Optimal Routing Cost. In *Proceedings of the 23rd Canadian Conference on Computational Geometry (CCCG '11)*. University of Toronto Press, 2011. to appear. 67

[KKK$^+$11b] Mong-Jen Kao, Bastian Katz, Marcus Krug, D.T. Lee, Ignaz Rutter, and Dorothea Wagner. The Density Maximization Problem in Graphs. In Ding-Zhu Du and Bin Fu, editors, *Proceedings of the 17th Annual International Conference on Computing Combinatorics (COCOON'11)*, Lecture Notes in Computer Science. Springer-Verlag, 2011. to appear. 89

[KKL$^+$11] Rolf Klein, Marcus Krug, Elmar Langetepe, D.T. Lee, and Dorothea Wagner. Constructing Optimal Shortcuts in Directed Weighted Paths and Trees. In *Proceedings of the 27th European Workshop on Computational Geometry (EuroCG'11)*, 2011. 67

[KKP93] David R. Karger, Daphne Koller, and Steven J. Phillips. Finding the hidden path: time bounds for all-pairs shortest paths. *SIAM J. Comput.*, 22:1199–1217, December 1993. 87

[KKRW09] Bastian Katz, Marcus Krug, Ignaz Rutter, and Alexander Wolff. Manhattan-Geodesic Point-Set Embeddability and Polygonization . Technical Report 2009-17, ITI Wagner, Karlsruhe Institute of Technology (KIT), 2009. 119

[KKRW10] Bastian Katz, Marcus Krug, Ignaz Rutter, and Alexander Wolff. Manhattan-Geodesic Embedding of Planar Graphs. In David Eppstein and Emden R. Gansner, editors, *Proceedings of the 17th International Symposium on Graph Drawing (GD'09)*, volume 5849 of *Lecture Notes in Computer Science*, pages 207–218. Springer-Verlag, 2010. 119

[KKS04] Atsushi Kaneko, Mikio Kano, and Kazuhiro Suzuki. Path coverings of two sets of points in the plane. In János Pach, editor, *Towards a Theory of Geometric Graphs*, volume 342, pages 99–111. American Mathematical Society, 2004. 170

[KKY00] Atsushi Kaneko, Mikio Kano, and Kiyoshi Yoshimoto. Alternating Hamilton cycles with minimum number of crossings in the plane. *Internat. J. Comput. Geometry Appl.*, 10(1):73–78, 2000. 171

[Kle08] Achim Klenke. *Probability Theory – A Comprehensive Course.* Springer-Verlag, Berlin, 2008. 25

[Klo94] Ton Kloks. *Treewidth, Computations and Approximations.* Lecture Notes Comput. Sci. Springer-Verlag, 1994. 99, 100

[KM11] Lukasz Kowalik and Marcin Mucha. 35/44-approximation for asymmetric maximum tsp with triangle inequality. *Algorithmica*, 59:240–255, 2011. 10.1007/s00453-009-9306-3. 3

[KMR97] D. Karger, R. Motwani, and G. Ramkumar. On approximating the longest path in a graph. *Algorithmica*, 18:82–98, 1997. 10.1007/BF02523689. 112

[Kn09] Arie Koster and Xavier Mu noz, editors. *Graphs and Algorithms in Communication Networks: Studies in Broadband, Optical, Wireless and Ad Hoc Networks.* Texts in Theoretical Computer Science. An EATCS Series. Springer-Verlag, 2009. 2

[KPT08] Jan Kynčl, János Pach, and Géza Tóth. Long alternating paths in bicolored point sets. *Discrete Mathematics*, 308(19):4315–4321, 2008. 170

[KS10] Victor Korolev and Irina Shevtsova. An improvement of the Berry-Esseen inequality with applications to Poisson and mixed Poisson random sums. *Scandinavian Actuarial Journal*, 2010. 31, 32

[Kur30] Kazimierz Kuratowski. Sur le Probleme des Courbes Gauches en Topologie. *Fundamenta Mathematicae*, 15:271–283, 1930. 11

[Kur04] Maciej Kurowski. A 1.235 lower bound on the number of points needed to draw all n-vertex planar graphs. *Information Processing Letters*, 92(2):95–98, 2004. 148

[KV03] Jeong Han Kim and Van H. Vu. Generating random regular graphs. In *STOC '03: Proc. 35th Ann. ACM Symp. Theory of Computing*, pages 213–222, New York, NY, USA, 2003. ACM. 24

[KvL84] Mark R. Kramer and Jan van Leeuwen. *VLSI Theory*, volume 2 of *Advances in Computing Research*, chapter The complexity of wire routing and finding minimum area layouts for arbitrary VLSI circuits, pages 129–146. JAI Press, 1984. 3

[KW01] Michael Kaufmann and Dorothea Wagner, editors. *Drawing Graphs: Methods and Models*, volume 2025 of *Lecture Notes Comput. Sci.* Springer-Verlag, 2001. 14

[KW02] Michael Kaufmann and Roland Wiese. Embedding vertices at points: Few bends suffice for planar graphs. *J. Graph Algorithms Appl.*, 6(1):115–129, 2002. 120, 148

[LC08] Hsiao-Fei Liu and Kun-Mao Chao. Algorithms for finding the weight-constrained k longest paths in a tree and the length-constrained k maximum-sum segments of a sequence. *Theor. Comput. Sci.*, 407(1-3):349–358, 2008. 92

[Lei80] Charles E. Leiserson. Area-efficient graph layouts. In *21st Ann. IEEE Symp. Foundations of Computer Science*, pages 270–281, Los Alamitos, CA, USA, 1980. IEEE Computer Society. 3

[Lew09] Ted G. Lewis. *Network Science: Theory and Applications.* Wiley, 1 edition, 2009. 3

[LJC02] Yaw-Ling Lin, Tao Jiang, and Kun-Mao Chao. Efficient algorithms for locating the length-constrained heaviest segments with applications to biomolecular sequence analysis. *J. Comput. Syst. Sci.*, 65(3):570–586, 2002. 92

[LLL09] D. T. Lee, Tien-Ching Lin, and Hsueh-I Lu. Fast algorithms for the density finding problem. *Algorithmica*, 53(3):298–313, 2009. 92

[LNN06] Hoong Chuin Lau, Trung Hieu Ngo, and Bao Nguyen Nguyen. Finding a length-constrained maximum-sum or maximum-density subtree and its application to logistics. *Discrete Optimization*, 3(4):385–391, 2006. 95

[Lok09] Daniel Lokshtanov. *New Methods in Parameterized Algorithms and Complexity.* PhD thesis, University of Bergen Norway, 2009. 92

[LV08] Ming Li and Paul M.B. Vitnyi. *An Introduction to Kolmogorov Complexity and Its Applications.* Springer-Verlag, New York, 3 edition, 2008. 198

[Mar04] Dániel Marx. Eulerian disjoint paths problem in grid graphs is NP-complete. *Discrete Appl. Math.*, 143(1-3):336–341, 2004. 122

[MB93] William A. Mackaness and Kate M. Beard. Use of graph theory to support map generalization. *Cartography and Geographic Information Science*, 20:210–221, 1993. 198

[McC07] James D. McCabe. *Network Analysis, Architecture, and Design.* The Morgan Kaufmann Series in Networking. Morgan Kaufmann, 3 edition, 2007. 3

[MELS95] Kazuo Misue, Peter Eades, Wei Lai, and Kozo Sugiyama. Layout adjustment and the mental map. *Journal of Visual Languages & Computing*, 6(2):183–210, 1995. 198

[MN90] Kurt Mehlhorn and Stefan Näher. Dynamic fractional cascading. *Algorithmica*, 5:215–241, 1990. 10.1007/BF01840386. 142, 144

[MRS$^+$98] Madhav V. Marathe, R. Ravi, Ravi Sundaram, S. S. Ravi, Daniel J. Rosenkrantz, and Harry B. Hunt. Bicriteria network design problems,. *Journal of Algorithms*, 28(1):142–171, 1998. 91

[MRS07] William A. Mackaness, Anne Ruas, and L. Tiina Sarjakoski, editors. *Generalisation of Geographic Information.* Cartographic Modelling and Applications. Elsevier B.V., 2007. 198

[MSU06] Criel Merino, Gelasio Salazar, and Jorge Urrutia. On the length of longest alternating paths for multicoloured point sets in convex position. *Discrete Mathematics*, 306(15):1791–1797, 2006. 195

[MTB76] Gabriel Macaya, Jean-Paul Thiery, and Giorgio Bernardi. An approach to the organization of eukaryotic genomes at a macromolecular level. *Journal of Molecular Biology*, 108(1):237–254, 1976. 92

[MW11] Sascha Meinert and Dorothea Wagner. An Experimental Study on Generating Planar Graphs. In Mikhail Atallah, Xiang-Yang Li, and Binhai Zhu, editors, *Proc. 7th Internat. Conf. Algorithmic Aspects in Information and Management (AAIM'11)*, volume 6681 of *Lecture Notes Comput. Sci.*, pages 375–387. Springer-Verlag, 2011. 20

[Nag00] Hiroshi Nagamochi. Recent development of graph connectivity augmentation algorithms. *IEICE Trans. Inf. And Syst.*, E83-D(3):372–383, 2000. 68

[New10] Mark Newman. *Networks: An Introduction.* Oxford University Press, 1 edition, 2010. 3

[Nie06] Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms.* Oxford University Press, 2006. 17

[Nut09] Zeev Nutov. Approximating connectivity augmentation problems. *ACM Trans. Algorithms*, 6:5:1–5:19, December 2009. 68

[O'R88] Joseph O'Rourke. Uniqueness of orthogonal connect-the-dots. In G.T. Toussaint, editor, *Computational Morphology*, pages 97–104. North-Holland, 1988. 121, 122, 126

[osm11] Openstreetmap database. online, 2011. `http://www.openstreetmap.de/`. 216, 221, 226

[OvL81] Mark H. Overmars and Jan van Leeuwen. Maintenance of configurations in the plane. *Journal of Computer and System Sciences*, 23(2):166–204, 1981. 95

[PM11] Chutima Prommak and Sujitra Modhirun. Optimal wireless sensor network design for efficient energy utilization. In *Proc. 2011 IEEE Workshops of Internat. Conf. Advanced Information Networking and Applications*, WAINA '11, pages 814–819, Washington, DC, USA, 2011. IEEE Computer Society. 3

[PT02] János Pach and Géza Tóth. Monotone drawings of planar graphs. In Prosenjit Bose and Pat Morin, editors, *Proc. 13th Internat. Symp. Algorithms Comput. (ISAAC'02)*, volume 2518 of *Lecture Notes Comput. Sci.*, pages 647–654. Springer-Verlag, 2002. 120

[PTVF07] William H. Press, Saul A. Teukolsky, William T. Veterling, and Brian P. Flannery. *Numerical Recipes: The Art of Scientific Computing*, chapter 5.12 Padé Approximants, pages 245–246. Cambridge University Press, New York, 3 edition, 2007. 38

[PW01] János Pach and Rephael Wenger. Embedding planar graphs at fixed vertex locations. *Graph. Combinator.*, 17(4):717–728, 2001. 121

[QE01] Aaron Quigley and Peter Eades. Fade: Graph drawing, clustering, and visual abstraction. In J. Marks, editor, *Graph Drawing*, volume 1984 of *Lect. Not. Comput. Sci.*, pages 77–80. Springer-Verlag, Berlin / Heidelberg, 2001. 198

[Qua60] Richard E. Quandt. Models of transportation and optimal network concstruction. *Journal of Regional Science*, 2(1):27–45, 1960. 3

[Rap86] David Rappaport. On the complexity of computing orthogonal polygons from a set of points. Technical Report SOCS-86.9, McGill University, Montréal, 1986. 120, 122, 126

[RC05] Davood Rafiei and Stephen Curial. Effectively visualizing large networks through sampling. In *Visualization, 2005. VIS 05. IEEE*, pages 375–382, October 2005. 198

[RCS86] Raghunath Raghavan, James Cohoon, and Sartaj Sahni. Single bend wiring. *J. Algorithms*, 7(2):232–257, 1986. 121

[RGGO11] César Rego, Dorabela Gamboa, Fred Glover, and Colin Osterman. Traveling salesman problem heuristics: Leading methods, implementations and latest advances. *European Journal of Operational Research*, 211(3):427–441, 2011. 3

[rou] University of Oregon Routeviews Project. http://www.routeviews.org/. 216

[RS84] Neil Robertson and Paul D. Seymour. Graph minors. III. Planar tree-width. *Journal of Combinatorial Theory, Series B*, 36(1):49–64, 1984. 114

[RS95] Neil Robertson and Paul D. Seymour. Graph Minors XIII. The Disjoint Paths Problem. *Journal of Combinatorial Theory, Series B*, 63(1):65–110, 1995. 100

[RS04] Neil Robertson and Paul D. Seymour. Graph Minors. XX. Wagner's conjecture. *Journal of Combinatorial Theory, Series B*, 92(2):325–357, 2004. Special Issue Dedicated to Professor W.T. Tutte. 11

[RSM$^+$96] R. Ravi, R. Sundaram, M. V. Marathe, D. J. Rosenkrantz, and S. S. Ravi. Spanning trees – short or small. *SIAM J. Discret. Math.*, 9:178–200, May 1996. 116

[RW93] Franz Rendl and Gerhard Woeginger. Reconstructing sets of orthogonal line segments in the plane. *Discrete Math.*, 119(1-3):167–174, 1993. 121

[RW06] Ronald C. Read and Nicholas C. Wormald. Number of labeled 4-regular graphs. *Journal of Graph Theory*, 4(2):203–212, 2006. 24

[RW09] Frank Ruskey and Aaron Williams. The coolest way to generate combinations. *Discrete Mathematics*, 309(17):5305 – 5320, 2009. 60

[RZ00] Gabriel Robins and Alexander Zelikovsky. Improved steiner tree approximation in graphs. In *Proc. 11th Ann. ACM-SIAM Symp. Discrete Algorithms (Soda'00)*, SODA '00, pages 770–779, Philadelphia, PA, USA, 2000. Society for Industrial and Applied Mathematics. 92

[SA95] Micha Sharir and Pankaj K. Agarwal. *Davenport–Schinzel Sequences and their Geometric Applications.* Cambridge University Press, 1995. 79

[Saa95] Alan Saalfeld. Map generalization as a graph drawing problem. In Roberto Tamassia and Ioannis G. Tollis, editors, *Graph Drawing*, volume 894 of *Lect. Not. Comput. Sci.*, pages 444–451. Springer-Verlag, Berlin / Heidelberg, 1995. ix, 198

[Sar10] S. Bry Sarte. *Sustainable Infrastructure: The Guide to Green Engineering and Design.* Wiley, 1 edition, 2010. 2

[SB92] Manojit Sarkar and Marc H. Brown. Graphical fisheye views of graphs. In *Proc. SIGCHI Conf. Human Factors in Computing Systems*, CHI '92, pages 83–91, New York, NY, USA, 1992. ACM. 198

[Sch90] Walter Schnyder. Embedding planar graphs on the grid. In *Proc. 1st ACM-SIAM Symp. Discrete Algorithms (SODA'90)*, pages 138–148, 1990. 120

[Sch05] Alexander Schrijver. On the history of combinatorial optimization (till 1960). In George L. Nemhauser Karen Aardal and Robert Weismantel, editors, *Discrete Optimization*, volume 12 of *Handbooks in Operations Research and Management Science*, pages 1–68. Elsevier, 2005. 2

[Sco69] A. J. Scott. The optimal network problem: Some computational procedures. *Transportation Research*, 3(2):201–210, 1969. v, 3, 68

[Sco00] John P. Scott. *Social Network Analysis: A Handbook.* Sage Publications Ltd, 2 edition, 2000. 3

[Sei83] Stephen B. Seidman. Network Structure and Minimum Degree. *Social Networks*, 5:269–287, 1983. 22

[Sei88] Raimund Seidel. Constrained Delaunay triangulations and Voronoi diagrams with obstacles. Technical Report 260, IIG-TU Graz, Austria, 1988. 73

[SIN07] Xiao Zhou Shuji Isobe and Takao Nishizeki. Total colorings of degenerate graphs. *Combinatorica*, 27(2):167–182, March 2007. 22

[SP80] J. M. S. Simões-Pereira. Erdős-Hajnal well-orderings and n-degenerate graphs. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 50(1):101–107, December 1980. 22

[SW] Petra Schuurman and Gerhard Woeginger. Approximation schemes – a tutorial. preliminary version of a chapter in the book "Lectures on Scheduling", to appear in 2011. 103

[SW99] Angelika Steger and Nicholas C. Wormald. Generating random regular graphs quickly. *Comb. Probab. Comput.*, 8(4):377–396, 1999. 24

[Tam87] Roberto Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM J. Comput.*, 16(3):421–444, 1987. 123

[TE10] Alexandru Telea and Ozan Ersoy. Image-based edge bundles: Simplified visualization of large graphs. *Computer Graphics Forum*, 29(3):843–852, 2010. 198

[TNU09] Satoshi Tayu, Kumiko Nomura, and Shuichi Ueno. On the two-dimensional orthogonal drawing of series-parallel graphs. *Discrete Appl. Math.*, 157(8):1885–1895, 2009. 123

[Val81] Leslie G. Valiant. Universality considerations in VLSI circuits. *IEEE Trans. Comput.*, 30:135–140, February 1981. 3

[WC04] Bang Ye Wu and Kun-Mao Chao. *Spanning trees and optimization problems.* Chapman & Hall/CRC, Boca Raton, 2004. 68

[WCT99] Bang Ye Wu, Kun-Mao Chao, and Chuan Yi Tang. An efficient algorithm for the length-constrained heaviest path problem on a tree. *Inf. Process. Lett.*, 69(2):63–67, 1999. 92

[Wer68] Christian Werner. The role of topology and geometry in optimal network design. *Papers in Regional Science*, 21(1):173–189, 1968. 3

[WLB$^+$99] Bang Ye Wu, Giuseppe Lancia, Vineet Bafna, Kun-Mao Chao, R. Ravi, and Chuan Yi Tang. A polynomial-time approximation scheme for minimum routing cost spanning trees. *SIAM J. Comput.*, 29:761–778, 1999. 68

[Won80] Richard T. Wong. Worst-case analysis of network design problem heuristics. *SIAM Journal on Algebraic and Discrete Methods*, 1(1):51–63, 1980. 68

[Wor99] Nicholas C. Wormald. Models of random regular graphs. In *Surveys in Combinatorics*, pages 239–298. Cambridge University Press, 1999. 24

[Wu09] Bang Ye Wu. An optimal algorithm for the maximum-density path in a tree. *Inf. Process. Lett.*, 109(17):975–979, 2009. 92, 95

Index

List of Publications

Journal Articles

[1] **Augmenting k-Core Generation with Preferential Attachment**. *Networks and Heterogeneous Media*, 3(2):277–294, June 2008. Joint work with Michael Baur, Marco Gaertler, Robert Görke, and Dorothea Wagner. 22

Articles in Refereed Conference Proceedings

[2] **Generalizing Geometric Graphs**. In: *Proceedings of the 19th International Symposium on Graph Drawing (GD'11)*, Lecture Notes in Computer Science. Springer-Verlag, 2012. to appear. Joint work with Edith Brunel, Andreas Gemsa, Ignaz Rutter, and Dorothea Wagner. 197

[3] **Orthogeodesic Point-Set Embedding of Trees**. In: *Proceedings of the 19th International Symposium on Graph Drawing (GD'11)*, Lecture Notes in Computer Science. Springer-Verlag, 2012. to appear. Joint work with Emilio Di Giacomo, Fabrizio Frati, Radoslav Fulek, and Luca Grilli. 147

[4] **Hamiltonian Orthogeodesic Alternating Paths**. In: *Proceedings of the 22nd International Workshop on Combinatorial Algorithms*, Lecture Notes in Computer Science. Springer-Verlag, 2012. to appear. Joint work with Emilio Di Giacomo, Luca Grilli, Giuseppe Liotta, and Ignaz Rutter. 169

[5] **Orthogonal Graph Drawing with Flexibility Constraints**. In: Ulrik Brandes and Sabine Cornelsen, editors, *Proceedings of the 18th International Symposium on Graph Drawing (GD'10)*, volume 6502 of *Lecture Notes in Computer Science*, pages 92–104. Springer-Verlag, 2011. Joint work with Thomas Bläsius, Ignaz Rutter, and Dorothea Wagner.

[6] **Connecting Two Trees with Optimal Routing Cost**. In: *Proceedings of the 23rd Canadian Conference on Computational Geometry (CCCG '11)*. University of Toronto Press, 2011. to appear. Joint work with Mong-Jen Kao, Bastian Katz, D.T. Lee, Martin Nöllenburg, Ignaz Rutter, and Dorothea Wagner. 67

[7] **The Density Maximization Problem in Graphs**. In: Ding-Zhu Du and Bin Fu, editors, *Proceedings of the 17th Annual International Conference on Computing Combinatorics (COCOON'11)*, Lecture Notes in Computer Science. Springer-Verlag, 2011. to appear. Joint work with Mong-Jen Kao, Bastian Katz, D.T. Lee, Ignaz Rutter, and Dorothea Wagner. 89

[8] **Gateway Decompositions for Constrained Reachability Problems**. In: Paola Festa, editor, *Proceedings of the 9th International Symposium on Experimental Algorithms (SEA'10)*, volume 6049 of *Lecture Notes in Computer Science*, pages 449–461. Springer-Verlag, May 2010. Joint work with Bastian Katz, Andreas Lochbihler, Ignaz Rutter, Gregor Snelting, and Dorothea Wagner.

[9] **Preprocessing Speed-Up Techniques is Hard**. In: *Proceedings of the 7th Conference on Algorithms and Complexity (CIAC'10)*, volume 6078 of *Lecture Notes in Computer Science*, pages 359–370. Springer-Verlag, 2010. Joint work with Reinhard Bauer, Tobias Columbus, Bastian Katz, and Dorothea Wagner.

[10] **Synthetic Road Networks**. In: *Proceedings of the 6th International Conference on Algorithmic Aspects in Information and Management (AAIM'10)*, volume 6124 of *Lecture Notes in Computer Science*, pages 46–57. Springer-Verlag, 2010. Joint work with Reinhard Bauer, Sascha Meinert, and Dorothea Wagner.

[11] **Enumerating and Generating Labeled k-Degenerate Graphs**. In: *Proceedings of the 7th Workshop on Analytic Algorithmics and Combinatorics (ANALCO '10)*, pages 90–98. SIAM, 2010. Joint work with Reinhard Bauer and Dorothea Wagner. 19

[12] **Manhattan-Geodesic Embedding of Planar Graphs**. In: David Eppstein and Emden R. Gansner, editors, *Proceedings of the 17th International Symposium on Graph Drawing (GD'09)*, volume 5849 of *Lecture Notes in Computer Science*, pages 207–218. Springer-Verlag, 2010. Joint work with Bastian Katz, Ignaz Rutter, and Alexander Wolff. 119

[13] **Minimizing the Area for Planar Straight-Line Grid Drawings**. In: Seok-Hee Hong, Takao Nishizeki, and Wu Quan, editors, *Proceedings of the 15th International Symposium on Graph Drawing (GD'07)*, volume 4875 of *Lecture Notes in Computer Science*, pages 207–212. Springer-Verlag, January 2008. Joint work with Dorothea Wagner.

[14] **Generating Graphs with Predefined k-Core Structure**. In: *Proceedings of the European Conference of Complex Systems (ECCS'07)*, October 2007. Joint work with Michael Baur, Marco Gaertler, Robert Görke, and Dorothea Wagner. 24

Articles in Non-Refereed Workshop Proceedings

[15] **Constructing Optimal Shortcuts in Directed Weighted Paths and Trees**. In: *Proceedings of the 27th European Workshop on Computational Geometry (EuroCG'11)*, 2011. Joint work with Rolf Klein, Elmar Langetepe, D.T. Lee, and Dorothea Wagner. 67

Thesis

[16] **Minimizing the Area for Planar Straight-Line Grid Drawings**. Master's thesis, Universität Karlsruhe, 2007.