# Optimal Interconnection Trees in the Plane

*Theory, Algorithms and Applications*

Marcus Brazil Martin Zachariasen

*To Jacinta and Elin*

# Contents

# Preface

Physical networks are pervasive in our society. They range from the traditional copper networks in telecommunications to the modern optical cable networks used for broadband communications; from the microscopic networks of interconnect within a microchip to the invisible wireless networks used by sensors and mobile phones. Roads and rail are examples of networks, as are the systems of pipes for water distribution, or the systems of high and low voltage cables for power distribution. All of these networks are vital for the smooth functioning of our society, but they are often costly from either an economic or environmental point of view. This book studies the fundamental problem of how we can design networks of these types that are functional but as cost-efficient as possible.

There are two essential aspects to efficient network design: the combinatoric and the geometric. The combinatoric aspect is about making decisions which involve choices from a finite (but possibly extremely large) set of possibilities. For example, if we define the topology of a network as the pattern of interconnections within the network or, equivalently, as its underlying graph structure, then the problem of determining the correct topology for an optimal network is combinatoric. The geometric aspect is about those design decisions which rely on the mathematical properties of the space in which the network sits, and which generally range over a continuous set of possibilities. For example, finding the optimal location of a junction or relay within a network is a geometric problem.

Although there are many books on network optimisation, most focus exclusively on the combinatorial aspects of the problem, using a wealth of techniques from graph theory and operations research. Such an approach works particularly well in cases where the network is constrained to fitting into a given physical infrastructure, such as multicast routing over the Internet. There are, however, very few books that consider, in any depth, the geometric aspects of optimal network design, and none that covers the exciting advances in the field over the past fifteen to twenty years. The aim of this book is to fill this gap in the literature in an accessible and illuminating way.

This book explores some of the fundamental aspects of geometric network optimisation with applications to a range of real-world problems. At the simplest level these problems involve taking a given set points at known locations in a metric space, and designing a network that interconnects the points while minimising a given cost function. It is assumed that this cost function depends on aspects of the geometry of the network, such as the lengths and embeddings of arcs. Without any further constraints, such optimal networks are almost always trees, i.e., networks containing no cycles. For this reason, we restrict ourselves in this book to properties of trees. We also assume that although an optimal tree must interconnect, and hence contain as nodes, a given set of embedded points, it may also contain other nodes whose potential locations are essentially unconstrained. These extra, variable nodes are usually referred to as Steiner points (after the nineteenth century mathematician, Jacob Steiner), and the problem of constructing such optimal trees is known as the (geometric) Steiner tree problem. The nature of the Steiner tree problem depends very much on the underlying metric space and the definition of the cost function.

In this book we focus principally on the geometric structure of optimal Steiner trees in the plane, under a range of metrics and cost functions, and on how this understanding of the structure can be used to design practical algorithms for constructing exact solutions to the Steiner tree problem. Almost all Steiner tree problems are known to be intrinsically difficult, in the sense that they belong to the class of NP-hard problems, a class for which polynomial-time solutions are believed not to exist. Nevertheless, the strongly constrained geometric structure exhibited by minimum Steiner trees in many metrics makes it possible to design algorithms for exactly solving the Steiner tree problem that often perform extremely well in practice.

Our decision to restrict the focus of this book to exact algorithms reflects our interest in understanding the mathematical structure of optimal Steiner trees in as much depth as possible. One of the great breakthroughs in this area over the last twenty years has been the development of the GeoSteiner optimisation tool at the University of Copenhagen. By cleverly exploiting both local and global structural properties of minimum Steiner trees, this software is able to exactly solve the Steiner tree problem for thousands of given points in a range of metric environments. GeoSteiner shows the power of rigorous mathematical analysis in helping design fast practical algorithms that achieve the best results possible. Of course we recognise that heuristics and approximation algorithms also have an important role (and indeed may be employed by exact algorithms to provide good upper bounds), but a detailed discussion of these sorts of techniques is left for another place.

One of the important aims of this book is to bring together the somewhat disparate literature on Steiner trees into a common framework to build a cohesive mathematical theory for the area. In doing so, we have tried to develop a consistent language and viewpoint for describing the theory for different metrics and constraints. Where possible we have also tried to draw useful connections between related topics, such as using the general canonical forms for minimum Steiner trees under fixed orientation metrics to better understand the well-known Hwang form for minimum rectilinear Steiner trees, to give just one example.

## Structure and features of the book

This book consists of four main chapters on geometric Steiner tree problems, followed by a short final chapter on the Steiner tree problem in graphs. Chapter 1 begins with a comprehensive study of the Euclidean Steiner tree problem. This is the most intuitively appealing of the Steiner tree problems and the one with the longest history. The first two sections of this chapter present the fundamental theory, which then leads into topics including computational complexity, algorithmic approaches and special cases. The chapter then looks at how some of the fundamental properties of Euclidean Steiner trees can be generalised to arbitrary norms, establishing some key general properties which are drawn on in the later chapters.

Chapter 2 covers the comparatively recent theory of Steiner trees for fixed orientation metrics. This is equivalent to solving the Steiner tree problem under a norm in which the unit ball is a centrally symmetric polygon. The chapter builds on some of the more general properties of normed Steiner trees from Chapter 1. Here the fundamental theory is developed in the first three sections, followed by discussions of global properties and algorithms.

The rectilinear Steiner tree problem, which is the most important form of the Steiner tree problem in terms of current applications (mostly involving the design of microchips), is the subject of Chapter 3. We have attempted to make this chapter reasonably self-contained, although a familiarity with some of the concepts in Chapter 2, such as direction sets and canonical forms, is assumed. Here the fundamental properties are described in the first section, followed by global properties, two different algorithmic approaches and a discussion of special sets of points for which polynomial-time algorithms exist.

Chapter 4 features four variants of the Steiner problem with cost functions other than those discussed in the first three chapters, and in some cases with extra constraints. Each of these variants is interesting from both a theoretical and an applications point of view. The main topics covered in this chapter are: the gradient-constrained Steiner tree problem, which is another example of a Steiner tree problem in a normed plane with interesting properties and applications; the obstacle-avoiding Steiner tree problem, in a comprehensive new treatment; Steiner tree problems in which there is a fixed bound on the number of Steiner points, including the geometric bottleneck Steiner tree problem; and problems involving Steiner trees minimising flow costs, such as Gilbert trees. The chapter concludes with a brief discussion of some related problems.

Finally, Chapter 5 is a brief survey of some of the main properties and algorithms connected with the Steiner tree problem in graphs and hypergraphs. The problems in this chapter are somewhat different from those in the rest of the book, as they are purely combinatorial. This chapter has been included for completeness and because some of the results on hypergraphs are crucial ingredients in the concatenation phase of the GeoSteiner algorithm discussed in the earlier chapters.

Some of the other features of this book include:

- **Sections on applications and extensions.** Each chapter, or in the case of Chapter 4 each main section, concludes with a section in which we discuss a range of real-world and mathematical applications, as well as extensions. The extensions include additional constraints and generalisations to higher dimensions.

- **Colour figures.** Many of the geometrical concepts underlying the mathematical formulations and proofs in this book are highly visual in nature. To make the mathematical ideas as accessible as possible, we have used a consistent visual language in the figures, and have made extensive use of colour to help clarify the concepts and constructions. We are grateful to our publisher, Springer, for supporting our use of colour printing throughout the book.

- **Exercises.** This book is suitable for use as a teaching text at a graduate level. To help facilitate this, we have included exercises at the end of each chapter. All exercises are referenced in the main text.

- **End notes.** Each chapter has a number of notes at the very end. Many of these relate to various aspects of the history of the Steiner tree problem.

- **Index.** The book has a comprehensive index. Note, however, that the index does not record every instance of a term or concept, but rather points to the first introduction or definition of a concept (or in a few cases also to a significant reintroduction). In this way, the index acts more as a sort of glossary than as a conventional index.

## Background and supplementary reading

One of the great attractions of the Steiner tree problem is that it is easy to state and understand but very challenging to solve. Nevertheless, most variants of the problem can be effectively tackled using classical techniques from computational geometry, combinatorics and discrete mathematics. We assume the reader of this book has a good undergraduate background in mathematics, including areas such as graph theory, discrete mathematics and metric spaces, and in computer science, including topics such as computational complexity and algorithm design. Beyond these background requirements, we have attempted to make the book reasonably self-contained.

Some useful background textbooks include: *Introduction to Graph Theory*, by West [399]; *Metric Spaces*, by Searcóid [345]; *Introduction to Algorithms*, by Cormen et al. [121]; and the classic book on computational complexity, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, by Garey and Johnson [171]. For those who wish to know more about Minkowski spaces, we recommend the book, *Minkowski Geometry*, by Thompson [370], and for a comprehensive treatment of combinatorial optimisation we suggest *Combinatorial Optimization: Theory and Algorithms*, by Korte and Vygen [239].

For supplementary reading, we suggest the following books and monographs: *The Steiner Tree Problem*, by Hwang et al. [213] — this was the first major study of geometric and combinatorial Steiner tree problems, but is now more than twenty years old; *On Optimal Interconnections for VLSI*, by Kahng and Robins [231], which describes, from a geometric perspective, algorithms for high-performance, high-density interconnections during the routing phases of circuit layout, including heuristics for the rectilinear Steiner problem; *The Steiner Tree Problem: A Tour Through Graphs, Algorithms, and Complexity*, by Prömel and A. Steger [319], which includes a detailed treatment of the Steiner tree problem in graphs; *Geometric Spanner Networks*, by Narasimhan and Smid [293], on the closely related network optimisation problem of constructing geometric spanners; and *Steiner Tree Problems in Computer Communication Networks*, by Du and Hu [135], which considers a range of Steiner tree problems, though almost exclusively from a combinatorial point of view.

**Acknowledgements**

The authors wish to thank all of the many people associated with the production of this book. These include:

- Ron Graham and Jens Vygen, who encouraged and supported the project. Both were official opponents of the second author's Doctor of Science Dissertation, 'Fixed Orientation Interconnection Problems: Theory, Algorithms and Applications' [433], and both urged him to convert the dissertation into a 'real book' about the Steiner tree problem and related research, which, after two years of work, has resulted in this current volume. We are also grateful to Jens Vygen for inviting us to visit the Research Institute for Discrete Mathematics, University of Bonn, for three weeks in 2012.

- Stephan Held, Stefan Hougardy, Tim Nieberg, Dirk Müller and others at the Research Institute for Discrete Mathematics, University of Bonn, for helpful discussions on VLSI design optimisation and for providing feedback on our sections on applications to microchip design.

- Pawel Winter, David Warme, Rasmus Fonseca, Daniel Juhl, Ray Booth, Christian Wulff-Nilsen and a number of anonymous referees at Springer, for reading and providing helpful comments on early drafts of the book.

- Pimin Kefaloukos for helping produce Figure 1.6. Almost all of the other figures in this book were constructed by the authors using the IPE extensible drawing software developed by Otfried Cheong.

- Martin Peters, Ruth Allewelt, Torrey Adams and the editorial team at Springer, who have always been enthusiastic, patient and supportive.
- Our many Steiner colleagues whose collaborations with us, both published and unpublished, we have drawn upon in this book. These colleagues include: Pawel Winter, Hyam Rubinstein, Doreen Thomas, Nick Wormald, David Warme, Benny Nielsen, Marcus Volz, Konrad Swanepoel, Charl Ras and many others. We pay special tribute to our esteemed colleague and friend Jia Weng, who passed away in March 2014, having just celebrated his 76th birthday.
- Our families: Jacinta, Reuben and Natasha (in Melbourne), and Elin (in Copenhagen).

The first author also wishes to thank the Department of Computer Science DIKU, University of Copenhagen, which hosted a visit to Denmark for three months in 2012 and another one-month visit in 2014. Some of the work on this book was supported by a Discovery Grant from the Australian Research Council and by a grant awarded under the Melbourne Research Grant Support Scheme from the University of Melbourne.

Marcus Brazil
*Department of Electrical and Electronic Engineering,*
*The University of Melbourne,*
*Parkville, 3010, Australia*

Martin Zachariasen
*Department of Computer Science (DIKU),*
*University of Copenhagen,*
*Universitetsparken 5,*
*DK-2100 København Ø, Denmark*

*Symbols*

| | |
|---|---|
| $\lvert\cdot\rvert$ | Euclidean norm |
| $\lVert\cdot\rVert$ | Minkowski norm |
| $\mathbb{R}$ | Set of real numbers |
| $\mathbb{R}^2$ | Real number plane |
| $(p_x, p_y)$ | Cartesian coordinates of point $p$ |
| $pq$ | Straight line segment with endpoints $p$ and $q$ |
| $\overline{pq}$ | Straight line defined by points $p$ and $q$ |
| $\overrightarrow{pq}$ | Ray from $p$ passing through $q$ |
| $\overrightarrow{pq}$ | Vector from $p$ to $q$ |
| $\widehat{pq}$ | The Steiner arc of $p$ and $q$ |
| $\angle abc$ | Acute angle between $\overrightarrow{ab}$ and $\overrightarrow{bc}$ |
| $:=$ | Equals by definition |
| $\delta_G(v)$ | The neighbouring vertices to $v$ in a given graph $G$ |
| $\Omega$ | Set of obstacles |
| $V_\Omega$ | Set of convex vertices of all polygonal obstacles in $\Omega$ |
| conv | Convex hull of a region or set of regions |

# Chapter 1

# Euclidean and Minkowski Steiner Trees

The central question addressed throughout this book is that of how to interconnect a set of points in the plane in an optimal way. There are many ways of defining the 'optimality' of an interconnection, but the most familiar, from a geometric point of view, is Euclidean length. The very simplest version of this problem is as follows: How do we connect two points in the Euclidean plane such that the length of the connection is as short as possible? The answer was known by the ancient Greeks and is known by every schoolchild today: draw a straight line between the points.

Now suppose that we have three or more points in the plane. How do we create a shortest possible network that interconnects the points? This is the theme of this chapter. More specifically, we first consider the Euclidean problem, where length is measured in the usual way. At the end of the chapter we present some fundamental properties of shortest networks under a more general class of metrics: the Minkowski metrics.

## 1.1 Euclidean Steiner trees and local properties

### 1.1.1 The Fermat-Torricelli problem

We begin with a simple example. Suppose that $a$, $b$ and $c$ are the vertices of an equilateral triangle in the plane with side length $1$ as shown in Figure 1.1 (left). What is the shortest network interconnecting these vertices? That is, we wish to find an interconnection network such that the sum of the lengths of the edges of the network is minimum. One approach, shown in Figure 1.1 (centre), is to take the two shortest edges of the triangle, which in this case means any two edges of the equilateral triangle, resulting in a network of length $2$. The optimal solution, however, involves adding an extra vertex located at the centre of the equilateral triangle to the network, as shown in Figure 1.1 (right). Each edge

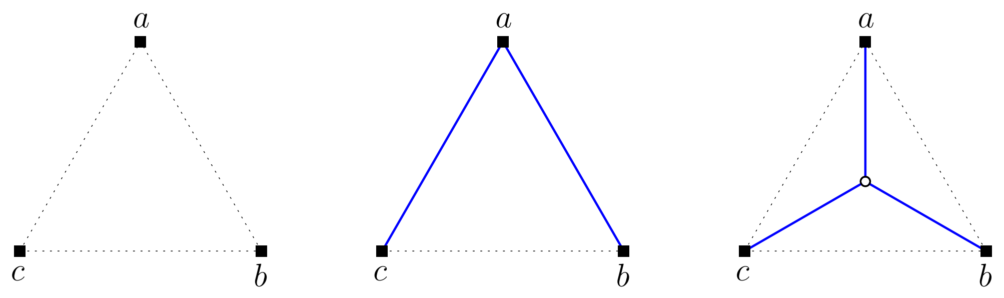


Figure 1.1: The vertices of an equilateral triangle (left) and two possible interconnecting networks (middle and right). The network on the right is the minimum length network where the possibility of including extra vertices is allowed.

from the central point to one of the vertices has a length that is $2/3$ of the height $\sqrt{3}/2$ of the equilateral triangle; i.e., each edge is of length $\sqrt{3}/3$. Hence, the network has total length $\sqrt{3}$ (which is less than 2). We will confirm later in this section that this solution is the best possible.

The above example is a specific instance of the following problem, posed by the seventeenth century mathematician, Pierre de Fermat, and first solved by the Italian physicist and mathematician Evangelista Torricelli:[1]

FERMAT-TORRICELLI PROBLEM
**Given**: A set of three points $N = \{a, b, c\}$ lying in the plane.
**Find**: A point $s$ such that the sum of Euclidean distances from $s$ to $a$, $b$ and $c$ is minimised.

The connections from $s$ to $a$, $b$ and $c$ form a *star* $T$ of length $|as|+|bs|+|cs|$. (Recall that a *star* on $k+1$ vertices is a tree containing one vertex of degree $k$ and $k$ vertices of degree 1.) In this star, $s$ may coincide with one of the given points in $N$ (making one of the edges degenerate). It is easy to see that a star $T$ solving the Fermat-Torricelli problem is a shortest network interconnecting the points in $N$ (see Exercise 1.1).

**Definition [Steiner point]**: Given points $a$, $b$ and $c$ in the plane, a point $s$ solving the Fermat-Torricelli problem for the given points will be referred to as a *Steiner point*.[2]

We now examine a method for solving the Fermat-Torricelli problem using the so-called *rotation proof*.[3] Not only is this an extremely elegant construction, but it also introduces some of the key concepts that we require for solving the more general Steiner tree problem later in this chapter.

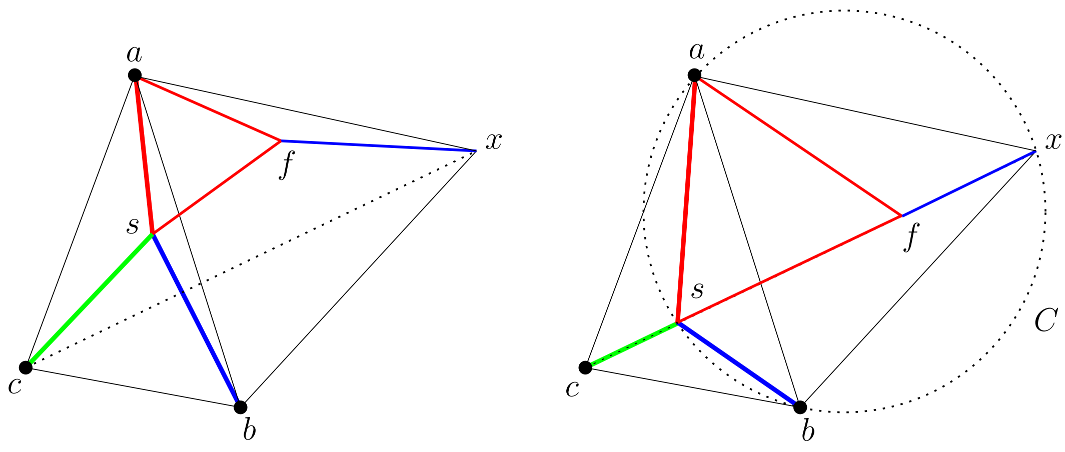


Figure 1.2: The rotation proof construction for $\triangle abc$. In each diagram, red, green or blue edges of the same colour have the same length. Hence, the sum of distances $|as| + |bs| + |cs|$ is equal to the length of the path $c - s - f - x$.

Consider three distinct points $a$, $b$ and $c$ in the plane. Assume, for now, that each angle in the triangle $\triangle abc$ formed by the points is less than $2\pi/3$ (or $120°$), and that the points $a$, $b$ and $c$ are labelled in clockwise order around $\triangle abc$. Suppose we wish to find a point $s$ solving the Fermat-Torricelli problem, i.e., such that $|as| + |bs| + |cs|$ is minimised. To this end, let $x$ be a point that is obtained by rotating $b$ counter-clockwise around $a$ through an angle of $\pi/3$. Point $x$ is the third point of an equilateral triangle with $ab$ as one of its sides (Figure 1.2, left).

Let $s$ be any arbitrary point in $\triangle abc$; let $f$ similarly be the point obtained by rotating $s$ counter-clockwise around $a$ through an angle of $\pi/3$. Note that $\triangle afs$ is equilateral. Since $x$ and $f$ are obtained via the same rotation around $a$, we have $|ab| = |ax|$, $|as| = |af|$ and $|sb| = |fx|$ (Figure 1.2, left). Our objective is to choose a point $s$ that minimises $|as| + |bs| + |cs| = |sf| + |fx| + |cs|$, which by reordering the terms is the same as $|cs| + |sf| + |fx|$. Therefore, if the path $c - s - f - x$ forms a straight line from $c$ to $x$, the minimum sum of distances is achieved.

To see that this minimum can be achieved, consider the circumcircle $C$ of the equilateral triangle $abx$. Let $s$ be the intersection (inside $\triangle abc$) of $C$ and $cx$ (Figure 1.2, right). Since $\angle asx$ and $\angle abx$ subtend the same chord of $C$, we have $\angle asx = \angle abx = \pi/3$. For this choice of $s$, point $f$ clearly lies on $cx$, so the minimum can indeed be achieved. Noting that $\angle bsx = \angle bax = \pi/3$, it also follows that $\angle asb = \angle bsc = \angle csa = 2\pi/3$.[4]

The same construction can be applied to the other two sides $bc$ and $ca$ of $\triangle abc$, using the following definitions and notation.

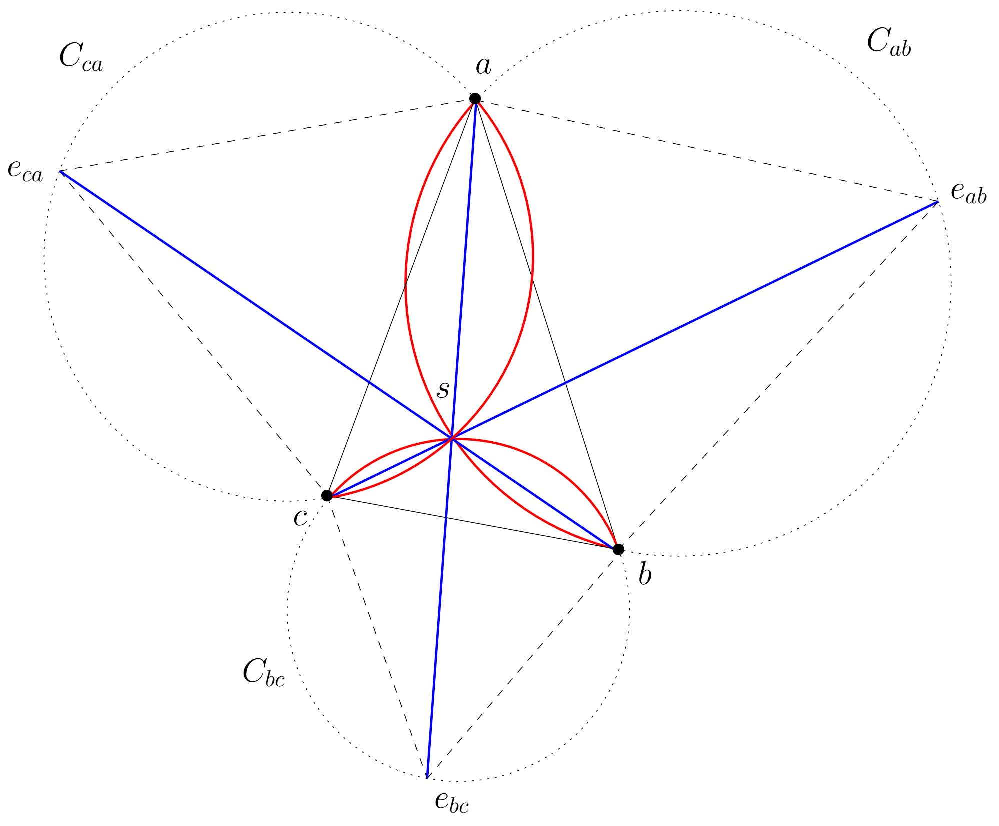


Figure 1.3: Equilateral points, Simpson lines (blue) and Steiner arcs (red) for $\triangle abc$.

**Definitions [Equilateral point, Simpson line]**: Given distinct points $a$ and $b$ in the plane, define the *equilateral point* $e_{ab}$ for $a$ and $b$ to be the third point of the equilateral triangle with vertices $a$ to $b$, such that the three vertices are labelled $a$, $b$ and $e_{ab}$ in counter-clockwise order around the triangle. Given three points $a$, $b$ and $c$ labelled in clockwise order around $\triangle abc$, the *Simpson line* for $c$ is the line segment $ce_{ab}$.

Note that in the example in Figure 1.2, we have $x = e_{ab}$. Then the Steiner point is also the unique intersection point of the three Simpson lines, $ae_{bc}$, $be_{ca}$ and $ce_{ab}$ (Figure 1.3). Furthermore, each Simpson line has length $|as| + |bs| + |cs|$, where $s$ is the Steiner point.

It is also clear from Figure 1.3 that we can construct the Steiner point using a slightly different method — called the Torricelli construction — by drawing the circumcircles of the three equilateral triangles on the sides of $\triangle abc$. The unique intersection point of these circles is also the Steiner point.

**Definition [Steiner arc]**: Given three points $a$, $b$ and $c$ labelled in clockwise order around $\triangle abc$, let $C_{ab}$ be the circumcircle of the equilateral triangle $abe_{ab}$. Then the open arc of $C_{ab}$ between $a$ and $b$ (and not containing $e_{ab}$) is called the *Steiner arc* of $a$ and $b$, denoted $\widehat{ab}$.

It now follows that the Steiner point can be obtained as the intersection of any pair of Simpson lines and/or Steiner arcs for $\triangle abc$.

Finally, if one of the angles of $\triangle abc$ is $2\pi/3$ or greater, it is easy to see that the point that minimises the sum of distances to $a$, $b$ and $c$ is equal to one of the given points — namely the one having an angle of $2\pi/3$ or greater (Exercise 1.2). In this case, the solution to the Fermat-Torricelli problem is the vertex shared by the two shortest edges of the triangle. The following theorem summarises the solution to the Fermat-Torricelli problem.

**Theorem 1.1** ***[Fermat-Torricelli problem solution]*** *Let $a$, $b$ and $c$ be three distinct points in the plane; let $s$ be the Steiner point minimising $|as| + |bs| + |cs|$.*

*(1) If each angle in the triangle $\triangle abc$ is less than $2\pi/3$, then*

*(i) $s$ can be obtained as the intersection between any pair of Simpson lines $ae_{bc}$, $be_{ca}$ and $ce_{ab}$ and/or Steiner arcs $\widehat{ab}$, $\widehat{bc}$ and $\widehat{ca}$;*

*(ii) the length of each Simpson line is $|as| + |bs| + |cs|$; and*

*(iii) the three angles around $s$ are each $2\pi/3$.*

*(2) If some angle in $\triangle abc$ at some vertex is greater than or equal to $2\pi/3$, then $s$ coincides with that vertex.*

It follows that a minimum length interconnection of three given points *either* consists of a star connecting a Steiner point to the three given points *or* consists of a pair of line segments connecting one of the given points to the two other points.

### 1.1.2 The Steiner tree problem

In this book we define a *geometric network* $G = (V(G), E(G))$ to be a connected graph that is embedded in the plane. The vertices $V(G)$ are points and the edges $E(G)$ are simple curves in the plane; let $|e|$ denote the Euclidean length of the embedding of edge $e \in E(G)$. If $s$ is a vertex of $G$, we define the *meeting angles* of $s$ to be the angles that appear counter-clockwise around $s$ in $G$. For example, if $s$ has three neighbouring vertices labelled $a$, $b$ and $c$ in counter-clockwise order around $s$, then the meeting angles are $\angle asb$, $\angle bsc$ and $\angle csa$ (see Figure 1.4).

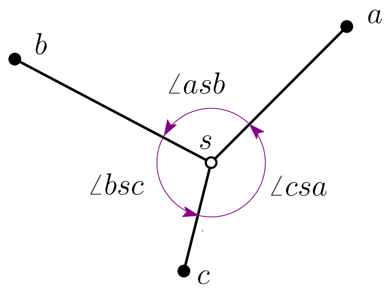


Figure 1.4: Meeting angles of a vertex $s$ in a geometric network.

We now describe the general problem considered in this chapter.[5]

EUCLIDEAN STEINER TREE PROBLEM IN THE PLANE
**Given**: A set of points $N = \{t_1, \dots, t_n\}$ lying in the plane.
**Find**: A geometric network $T = (V(T), E(T))$, such that $N \subseteq V(T)$, and such that $|T| := \sum_{e \in E(T)} |e|$ is minimised.

We make some observations concerning this definition. First of all we note that $T$, considered as a graph, can be assumed to be a tree: if $T$ contains a cycle, then any edge on the cycle can be removed without disconnecting the network or increasing $|T|$. Furthermore, all non-zero edges must be embedded as line segments, otherwise they would not have minimal length.

**Definitions [Minimum Steiner tree, terminals, Steiner points]**: A network $T = (V(T), E(T))$ representing a solution to the Steiner tree problem will be referred to as a *minimum Steiner tree*. The given points $N \subseteq V(T)$ are called *terminals*, and possible extra vertices $S = V(T) \setminus N$ are called *Steiner points* (as with the Fermat-Torricelli problem).[6]

All Steiner points in a minimum Steiner tree can be assumed to have degree at least 3 in $T$: a Steiner point of degree 1 and its incident edge can simply be removed, and a Steiner point of degree 2 can, together with its two incident edges, be replaced by a line segment between the opposite endpoints of the two incident edges. All these changes can be made without increasing the length of the tree.

Consider a vertex $a$ in $T$, and a pair of non-zero edges $ab, ac \in E(T)$ meeting at $a$. Note that $ab$ and $ac$ must form a shortest interconnection of $\{a, b, c\}$. As shown in the previous section, a solution to the Fermat-Torricelli problem for $\{a, b, c\}$ provides a shortest interconnection of $\{a, b, c\}$. An immediate consequence is that the angle $\angle bac$ must be at least $2\pi/3$. To see why, assume that $\angle bac < 2\pi/3$; then we have two cases as shown in Figure 1.5, and in neither case is $ab$ and $ac$ a shortest interconnection of $\{a, b, c\}$.

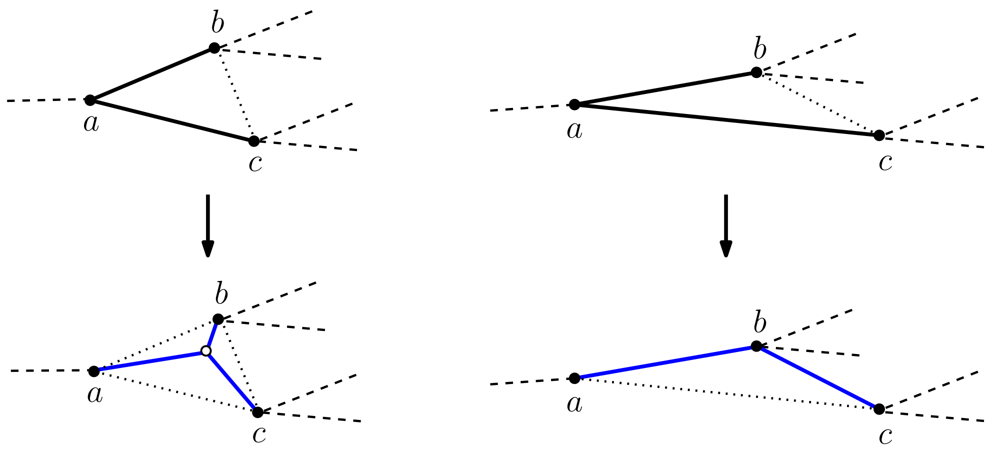


Figure 1.5: Replacing a pair of edges $ab$ and $ac$ for which $\angle bac < 2\pi/3$ with a shortest interconnection provided by a solution to the Fermat-Torricelli problem for $\{a, b, c\}$. On the left each angle in $\triangle abc$ is less than $2\pi/3$ (case (1) in Theorem 1.1). On the right one of the angles is greater than or equal to $2\pi/3$ (case (2) in Theorem 1.1). In both cases a shorter local tree is constructed.

As an immediate consequence of the fact that all meeting angles are at least $2\pi/3$, terminals have degree *at most 3* in $T$ and Steiner points have degree *exactly 3* in $T$. The following theorem summarises some basic properties of a minimum Steiner tree. The proofs of properties (1) and (4) are left as Exercises 1.5 and 1.6, respectively.

**Theorem 1.2** ***[Basic properties of a minimum Steiner tree]*** *For any finite set of points $N$ in the plane, there exists a minimum Steiner tree $T = (V(T), E(T))$ for terminals $N \subseteq V(T)$, having Steiner points $S = V(T)\backslash N$, satisfying the following four conditions:*

*(1) The edges of $T$ are line segments and are embedded in such a way that their interior does not intersect any other vertex or edge of $T$.*

*(2) Terminals in $N$ have degree at most 3 in $T$, and each pair of incident edges meets at an angle of $2\pi/3$ or greater.*

*(3) Steiner points in $S$ have degree 3 in $T$, and each pair of incident edges meets at an angle of $2\pi/3$.*

*(4) $T$ has at most $n - 2$ Steiner points and at most $2n - 3$ edges, where $n = |N|$ is the number of terminals.*

Figure 1.6 shows an example of a minimum Steiner tree. As noted above, the properties of a Steiner point in the Fermat-Torricelli problem (Theorem 1.1) transfer to each

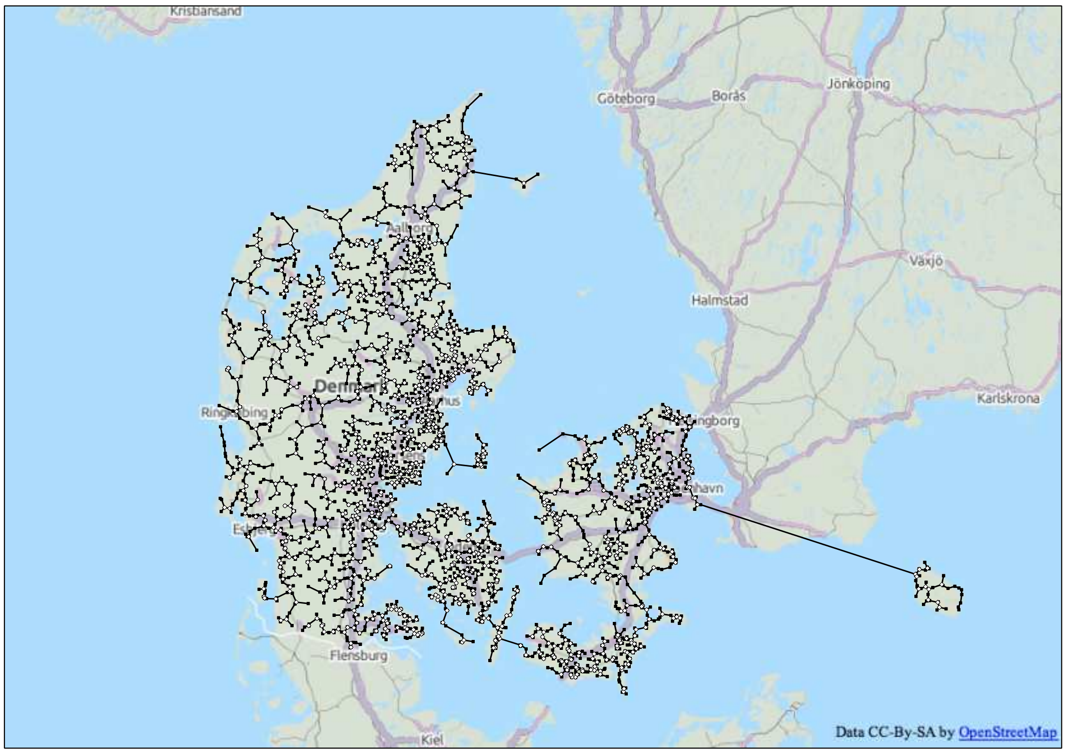


Figure 1.6: Euclidean minimum Steiner tree for 2741 cities in Denmark, found using GeoSteiner 4.0.

Steiner point in a minimum Steiner tree: a Steiner point $s$ with neighbours $a$, $b$ and $c$ must necessarily be a solution to the Fermat-Torricelli problem for $\{a, b, c\}$. Thus, if we know the location of neighbours $a$, $b$ and $c$, the Steiner point can be constructed in the manner described in Theorem 1.1.

### 1.1.3 Topologies and full components

The underlying graph $\mathcal{T} = (V(\mathcal{T}), E(\mathcal{T}))$ for a geometric network $T = (V(T), E(T))$ that interconnects a set of terminals $N$ is called the *topology* of $T$. In a topology, the vertices and edges are are not embedded in the plane, but each vertex is either labelled with the label of its corresponding terminal (having a given location in the plane) or labelled as a Steiner point (having no given location in the plane).

An embedding of a geometric network is called *non-degenerate* if all edges in the embedding have non-zero length — otherwise it is *degenerate*.

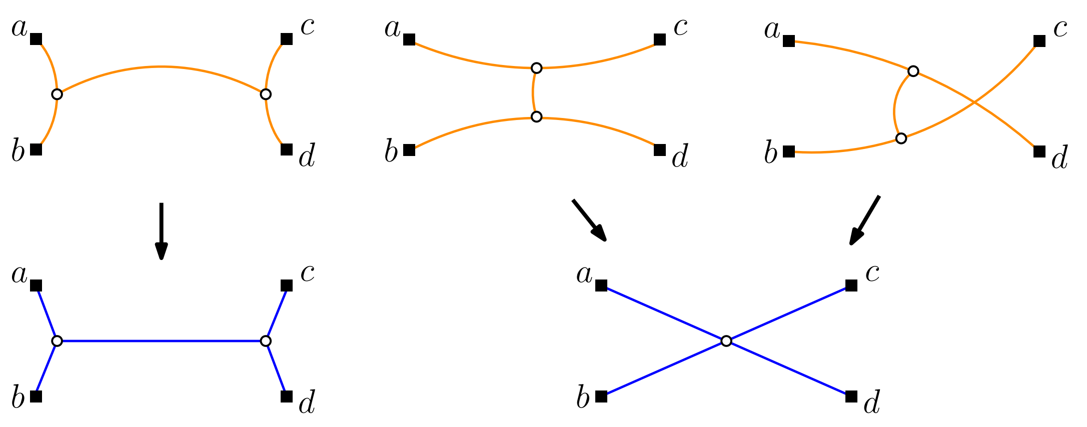


Figure 1.7: Three distinct full Steiner topologies (top row) and corresponding relatively minimal trees (bottom row) for a set of four terminals $\{a, b, c, d\}$. The three topologies are distinct, since terminals $a$ and $b$ have a common Steiner point as neighbour only in the topology on the left, whereas $a$ and $c$ have a common Steiner point as neighbour only in the middle topology. The relatively minimal tree on the left is a full Steiner tree, while the one on the right is not (since it contains a degenerate edge in the embedding).

**Definitions [Steiner topology, full Steiner topology]**: The topology of a non-degenerate minimum Steiner tree is called a *Steiner topology*. The topology of a non-degenerate minimum Steiner tree in which every terminal has degree 1 is called a *full Steiner topology*.

By Theorem 1.2 we can assume that Steiner vertices have degree exactly 3 in a Steiner topology; furthermore, terminals have degree at most 3 in a Steiner topology. A shortest possible embedding of a Steiner topology is called a *relatively minimal tree*; such an embedding is obtained by locating the Steiner vertices in such a way that the total length of the geometric network is minimised. A relatively minimal tree may be degenerate, in which case edges can have meeting angles that are less than $2\pi/3$ (Figure 1.7).[7]

**Definition [Steiner tree, full Steiner tree]**: A non-degenerate relatively minimal tree for a (full) Steiner topology $\mathcal{T}$ is called a *(full) Steiner tree* for $\mathcal{T}$.

Our interest in Steiner trees stems from the fact that there exists a Steiner tree that is a minimum Steiner tree; this follows from the definition of Steiner topologies and Steiner trees. Furthermore, a minimum Steiner tree that interconnects two or more terminals is a union of full Steiner trees (Figure 1.6); note that a full Steiner tree must contain at least one edge since each terminal has degree 1 in the tree. The following lemma characterises Steiner trees precisely.

**Lemma 1.3** ***[Necessary and sufficient properties of Steiner trees]*** *If a non-degenerate* *embedding $T$ of a Steiner topology $\mathcal{T}$ fulfils the four conditions of Theorem 1.2, then $T$ is the unique Steiner tree for $\mathcal{T}$.*

**Proof.** We prove this lemma in two steps. In Step 1 we show that if $T$ fulfils the conditions of Theorem 1.2, then $T$ is a Steiner tree. We then show, in Step 2, that such a $T$ is unique.

**Step 1.** Let $T$ be a non-degenerate embedding of a given Steiner topology $\mathcal{T}$. Let $L(T) := |T|$, the total edge length of $T$. For a fixed set of terminals we think of $L(T)$ as being a function of the locations of the Steiner points. Since the Euclidean length of an edge with respect to its endpoints is convex, it follows that $L(T) = \sum_{e \in E(T)} L(e)$ is convex with respect to the Steiner points $S$ of $T$ (since a sum of convex functions is convex). Hence, if $T$ is not a Steiner tree, then there exists a perturbation $\zeta$ of the elements of $S$ that reduces the length of $T$.

Now suppose $s_1$ and $s_2$ are adjacent Steiner points in $T$. Then $L(s_1s_2)$ is the length of the edge $s_1s_2$, and we denote by $\dot{L}(s_1s_2)_\zeta$ the derivative of this length with respect to $\zeta$. Let $\zeta_1$ and $\zeta_2$ represent the induced perturbations of $\zeta$ on $s_1$ and $s_2$ respectively (i.e., $\zeta_i = \zeta|_{s_i}$). We now show that

$$\dot{L}(s_1s_2)_\zeta = \dot{L}(s_1s_2)_{\zeta_1} + \dot{L}(s_1s_2)_{\zeta_2}. \tag{1.1}$$

Let $l$ be the line extending $s_1s_2$. Then $\dot{L}(s_1s_2)_\zeta$, $\dot{L}(s_1s_2)_{\zeta_1}$ and $\dot{L}(s_1s_2)_{\zeta_2}$ are each equal to the derivative of the length of the projection of $s_1s_2$ onto $l$ following the corresponding perturbation (since the component perpendicular to $l$ has derivative $0$), and Equation (1.1) immediately follows.

It follows from Equation (1.1) that $\dot{L}(s_1s_2)_\zeta < 0$ implies that either $\dot{L}(s_1s_2)_{\zeta_1} < 0$ or $\dot{L}(s_1s_2)_{\zeta_2} < 0$. By iterating this argument we conclude that if $\zeta$ is a length-reducing perturbation for $T$, then there exists a Steiner point $s \in S$ such that the induced perturbation on $s$ is also length-reducing. But if $T$ fulfils the conditions of Theorem 1.2, then no Steiner point of $T$ has a length-reducing perturbation, and hence $T$ is a Steiner tree.

**Step 2.** The Euclidean length of an edge with respect to its endpoints is convex, but not strictly convex; hence, we need a further argument to establish the uniqueness of a Steiner tree for a given topology. We argue by contradiction; assume there exist at least two distinct Steiner trees, $T_1$ and $T_2$, each with the same terminal set and topology. Since $T_1$ and $T_2$ are both relatively minimal trees with the same length, as we move the Steiner points of $T_1$ to their corresponding positions in $T_2$ (say, each at a constant rate), we create a continuum of relatively minimum intermediate trees between the two trees. Under this transformation from $T_1$ to $T_2$, there exists a Steiner point $s$ that is non-stationary but is adjacent to two stationary vertices (since all the terminals are fixed). However, in $T_1$, any sufficiently small movement of $s$ with respect to two fixed adjacent vertices means that $s$ no longer satisfies the condition that each pair of incident edges at $s$ meets at an angle of $2\pi/3$, contradicting the relative minimality of the intermediate trees between $T_1$ and $T_2$.

∎

If we split a Steiner topology $\mathcal{T}$ into separate connected components at each terminal that has degree more than 1, we obtain a set of *full components* that have full Steiner topologies. A full Steiner topology for $n = |N|$ terminals has exactly $n-2$ Steiner points. If a full Steiner tree exists for a given full Steiner topology, then it is unique (Lemma 1.3). From an algorithmic point of view, full Steiner topologies and full Steiner trees are particularly useful, since we are actually able to construct full Steiner trees from full Steiner topologies efficiently, as shown in the next section. The challenge is that the number of full Steiner topologies (and trees) increases rapidly as the number of terminals increases. A useful concept in the proof of this result, and throughout this chapter, is the following:

**Definition [Cherry]**: A pair of terminals adjacent to a common Steiner point in a Steiner topology is called a *cherry* for the topology.

Note that at least one cherry always exists for a full Steiner topology for $n \geq 3$ terminals (Exercise 1.7).

**Lemma 1.4** ***[Number of full Steiner topologies]*** *The number of distinct full Steiner topologies for $n$ terminals, where $n \geq 3$, is:*

$$f(n) = 1 \cdot 3 \cdot 5 \cdots (2n-7) \cdot (2n-5) = \frac{(2n-4)!}{(n-2)!\, 2^{n-2}}.$$

*Furthermore, each of these topologies can be realised by the unique minimum Steiner tree for some set of terminals.*[8]

**Proof.** A full Steiner topology interconnects $n$ terminals and $n-2$ Steiner points and hence has $2n-3$ edges. Let $f(n)$ be the number of full Steiner topologies spanning $n$ terminals. Clearly, $f(2) = f(3) = 1$. Assume that $\mathcal{T}_{n-1}$ is a full Steiner topology spanning $n-1$ terminals for $n \geq 3$. We can obtain a full Steiner topology on $n$ terminals by connecting the $n$th terminal $t_n$ to each edge $e$ of $\mathcal{T}_{n-1}$ by inserting a Steiner point on $e$ and connecting $t_n$ to this new Steiner point. Thus, we obtain a new full Steiner topology for each of the $2n-3$ edges in $\mathcal{T}_{n-1}$ (as illustrated for $n = 4$ in Figure 1.8). Conversely, in a full Steiner topology $\mathcal{T}_n$ with $n$ terminals, we can remove the $n$th terminal and its incident edge and make a shortcut at the Steiner point to create a full Steiner topology for $n-1$ terminals. The function $f(n)$ therefore fulfils the recursion

$$f(n) = (2(n-1)-3)f(n-1) = (2n-5)f(n-1)$$

from which the number of full Steiner topologies follows.

We next show that for each of the $f(n)$ full Steiner topologies, there exists a terminal set $N$ with $n = |N|$ such that the minimum Steiner tree for $N$ has that full Steiner topology. The proof is essentially by induction on the number of terminals.

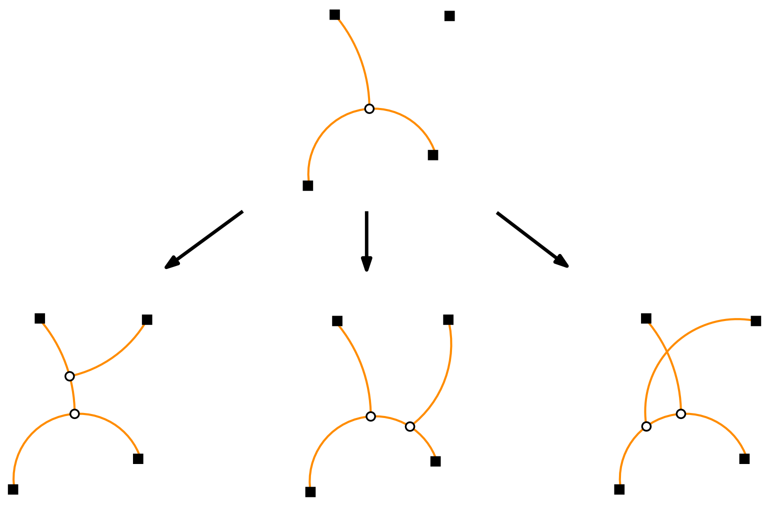


Figure 1.8: Possible full Steiner topologies for 4 terminals — based on a full Steiner topology for 3 terminals.

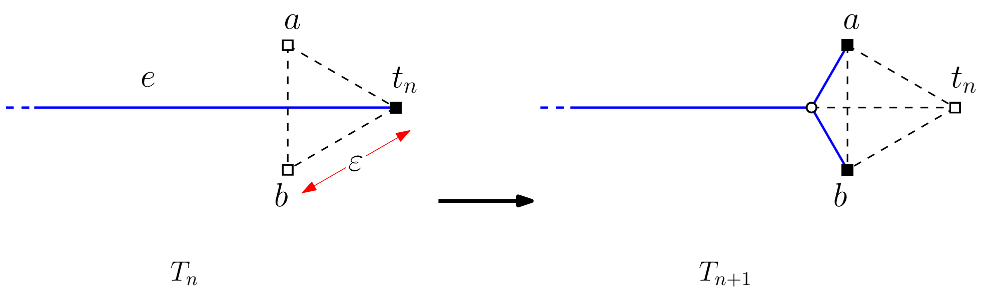


Figure 1.9: Given a minimum Steiner tree $T_n$ on $N$, we can replace a terminal $t_n$ by a pair of points $a$ and $b$ such that $e_{ab} = t_n$, as shown, resulting in a Steiner tree $T_{n+1}$ with terminal set $(N \setminus \{t_n\}) \cup \{a, b\}$ and with the same length as $T_n$.

Suppose there exists a full minimum Steiner tree $T_n$ with terminal set $N$, such that $|N| = n$, and such that $T_n$ is the unique minimum Steiner tree for $N$. Let $t_n$ be an arbitrary element of $N$, let $e$ be the edge of $T_n$ incident with $t_n$, and let $N_{n-1} := N \setminus \{t_n\}$. Since $T_n$ is unique, there exists a real number $\varepsilon > 0$ such that $|T'_n| - |T_n| > \varepsilon$ for every other relatively minimal tree $T'_n$ for $N$, and such that $\varepsilon \ll |e|$. We now define a pair of points $a$ and $b$ (designed to replace $t_n$) such that the line segment $ab$ satisfies the following properties: $ab \perp e$; the midpoint of $ab$ lies on $e$; the equilateral point $e_{ab} = t_n$; and $|ab| = |ae_{ab}| = |be_{ab}| = \varepsilon$ (Figure 1.9). It follows from the basic properties of Steiner trees, that by shortening the edge $e$ (at $t_n$) and adding two new edges connecting it to $a$ and $b$, as illustrated in Figure 1.9, we can construct a new Steiner tree $T_{n+1}$ with terminal set $N_{n-1} \cup \{a, b\}$ such that $|T_{n+1}| = |T_n|$.

We now show that $T_{n+1}$ is the unique minimum Steiner tree for $N_{n-1} \cup \{a, b\}$.

Suppose, to the contrary, there exists another Steiner tree $T'_{n+1}$ for $N_{n-1} \cup \{a, b\}$ (with different Steiner topology) such that $|T'_{n+1}| \leq |T_{n+1}|$. As before, we can remove either $a$ or $b$ from $T'_{n+1}$ and its Steiner topology, also removing its incident edge and making a shortcut at the adjacent Steiner point (if there is one), to create a Steiner tree $T'_a$ or $T'_b$ with terminal set $N_{n-1} \cup \{a\}$ or $N_{n-1} \cup \{b\}$, respectively. By construction, $|T'_a| \leq |T'_{n+1}|$ and $|T'_b| \leq |T'_{n+1}|$, and at least one of these Steiner trees, say $T'_a$, has Steiner topology different from that of $T_n$ (when $a$ is relabelled $t_n$). Now, consider the geometric network $T'_a \cup at_n$; this is a network of length $|T'_a| + \varepsilon$ interconnecting $N$. But, since $|T'_a| \leq |T_n|$, this implies that there exists a relatively minimal tree for $N$ of length at most $|T_n| + \varepsilon$ with different Steiner topology from $T_n$, giving a contradiction.

We can now iteratively repeat the above replacement method, starting with, say, the minimum Steiner tree for the vertices of an equilateral triangle, to create a minimum Steiner tree with any given full Steiner topology. ∎

The function $f(n)$ is super-exponential (Table 1.1), so enumerating all possible full Steiner topologies for large values of $n$ is infeasible. Many of the full Steiner topologies have, however, no associated full Steiner tree for a given configuration of terminals. For example, in Figure 1.8 we have $f(4) = 3$, but only two of these full Steiner topologies can lead to a full Steiner tree. Let us assume that we know the order in which the terminals appear if we make an outer walk of the corresponding full Steiner tree; that is, the circumferential order of the terminals is fixed. This could, for example, be the case when all terminals are vertices of their convex hull; here it can be shown (see Section 1.3.2) that the circumferential order must necessarily respect the order in which the terminals appear on the convex hull. If the circumferential order of a set of $n$ terminals is fixed, then the number of possible full Steiner topologies is the $(n-2)$th Catalan number

$$c(n-2) = \frac{(2n-4)!}{(n-1)!(n-2)!},$$

which grows asymptotically as $O(4^n)$; that is, the growth would (only) be exponential (Table 1.1). Figure 1.10 illustrates why the number of full Steiner topologies for a fixed circumferential order is the $(n-2)$th Catalan number: each full Steiner topology corresponds to $n-2$ applications of a binary operator, say, a complete parenthesisation of $n-1$ terminal labels, and it is well known that $c(n-2)$ is the number of different ways $n-1$ ordered symbols can be completely parenthesised. For more details, see [111]. It should be noted, however, that no general exponential bound is known on the number of possible (relevant) circumferential orders of the terminals, so in theory this does not reduce the number of possible full Steiner topologies to an exponential function of $n$.

| $n$ | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|
| $f(n)$ | 1 | 1 | 3 | 15 | 105 | 945 | 10395 | 135135 | 2027025 |
| $c(n-2)$ | 1 | 1 | 2 | 5 | 14 | 42 | 132 | 429 | 1430 |

Table 1.1: Number of full topologies $f(n)$ and Catalan numbers $c(n-2)$ for small values of $n$.

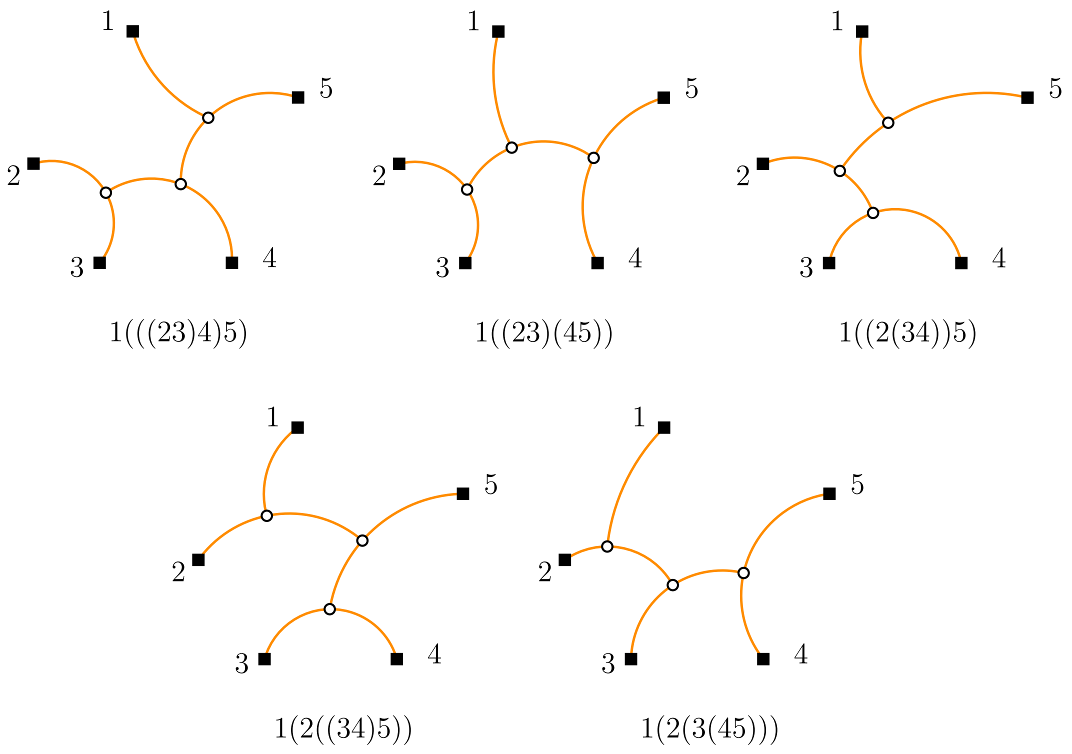


Figure 1.10: Correspondence between full Steiner topologies that respect a given circumferential order of the terminals, and parenthesisations of terminals labels. Note that terminal 1 does not participate in the parenthesisation.

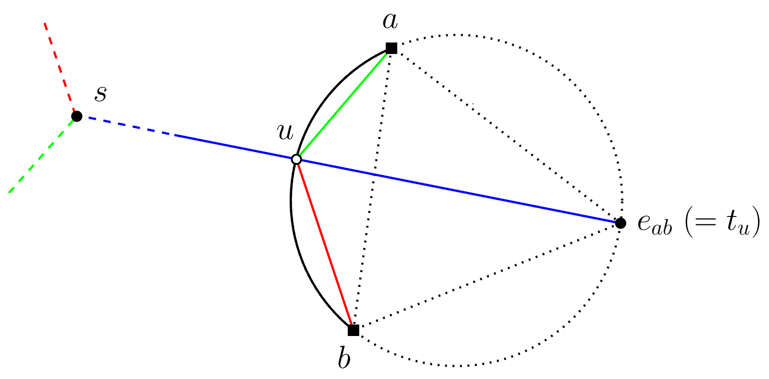


Figure 1.11: The basic recursive step in the Melzak-Hwang algorithm, involving replacing a cherry for a pair of terminals $a, b$ by a pseudo-terminal $t_u$.

## 1.2 Algorithms for a given Steiner topology

In this section we present an efficient algorithm — known as the Melzak-Hwang algorithm — for constructing a full Steiner tree for a given full Steiner topology (or deciding that it does not exist). Also, we address the problem of constructing a relatively minimal tree for an arbitrary, not necessarily full Steiner topology.

### 1.2.1 The Melzak-Hwang algorithm

Let $\mathcal{T}$ be a full Steiner topology for $n \geq 3$ terminals. In this section we give an efficient algorithm for computing a full Steiner tree $T$ for $\mathcal{T}$. First we present a basic recursive (and slow) algorithm, and then we show how the running time can be improved. Finally, we present an efficient implementation of the algorithm.

#### Basic recursive algorithm

Let $u$ be a Steiner point in $\mathcal{T}$ with neighbouring vertices $a, b$ and $s$, where $a$ and $b$ are terminals (and hence form a cherry). Denote by $\overline{ab}$ the straight line passing through $a$ and $b$. Let $e_{ab}$ be the equilateral point of $a$ and $b$, and assume that in the full Steiner tree $T$ for $\mathcal{T}$ the vertex $u$ is located on the opposite side of $\overline{ab}$ to $e_{ab}$. By Theorem 1.1, $u$ must lie on the Steiner arc $\widehat{ab}$ (on the same side of $e_{ab}$ as $s$) as illustrated in Figure 1.11; furthermore, both $e_{ab}$ and $\widehat{ab}$ can be constructed in constant time from $a$ and $b$.

It also follows from Theorem 1.1 that the Simpson line starting at $e_{ab}$ and passing through $u$ will overlap with edge $us$ in $T$, and that $|e_{ab}u| = |au| + |bu|$. In other words, not only does the Simpson line start at $e_{ab}$, but the total length of segments $au$ and $bu$ is identical to the length of the part of the Simpson line from $e_{ab}$ to $u$. Hence, we may replace $a$, $b$ and $u$ with a new point $t_u = e_{ab}$ that is connected directly to $s$. We refer to this new point as a *pseudo-terminal*, since it plays the role of a terminal in a simpler tree of equal length to $|T|$. (This is the reverse procedure to that illustrated in Figure 1.9.)

Let $\mathcal{T}'$ be the full Steiner topology on the new set of $n-1$ terminals (that is, the same set of terminals as for $\mathcal{T}$ but where $a$ and $b$ have been replaced by $t_u$), and let $T'$ be the full Steiner tree for $\mathcal{T}'$. Then the location of $u$ can be determined by taking the intersection between Steiner arc $\widehat{ab}$ and edge $t_u s$ in $T'$. The full Steiner tree $T$ for $\mathcal{T}$ is constructed by deleting edge $t_u s$ in $T'$, and adding the edges $au$, $bu$ and $us$. Note that $T$ exists if and only if $T'$ exists and the edge $t_u s$ in $T'$ intersects the Steiner arc $\widehat{ab}$.

The reduction from a problem on $n$ terminals to a problem on $n-1$ terminals is called a *merging* step. Constructing $T$ from $T'$ is called a *reconstruction* step. The complete algorithm therefore consists of $n-2$ merging steps (one for each Steiner point), and similarly $n-2$ reconstruction steps; after $n-2$ merging steps the topology consists of two terminals for which the full Steiner tree is a simple line segment.

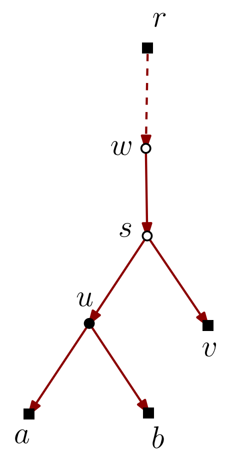
## Identifying the side of the equilateral point

The major difficulty in implementing the above algorithm is that for any pair of terminals $a, b$ forming a cherry we do not know on which side of $\overline{ab}$ to construct the pseudo-terminal $t_u$ in each merging step. There are two possibilities for $t_u$, corresponding to the two equilateral points $e_{ab}$ and $e_{ba}$; this creates two subproblems for each merging step — in total a running time of $O(2^n)$. We now show that we can in fact implement the algorithm to run in $O(n)$ time. This is achieved by selecting a particular merging order, which makes it possible to correctly identify a single equilateral point in each merging step.

For convenience assume that topology $\mathcal{T}$ has five or more terminals, that is, at least three Steiner points (otherwise, we solve the problem by brute force). Select an arbitrary terminal $r$ in $\mathcal{T}$, and denote by $\mathcal{T}_r$ the tree $\mathcal{T}$ with $r$ as its root. Note that every Steiner point in $\mathcal{T}_r$ is an internal node with two children. Let $u$ be a Steiner point that is furthest away from $r$ in $\mathcal{T}_r$ (or deepest in the rooted tree); therefore, $u$ has two terminals as children, which we denote by $a$ and $b$. Let $s$ be the parent of $u$, and let $v$ be the second child of $s$ (or the sibling of $u$). Finally, let $w$ be the parent of $s$.

Since $u$ is chosen as a Steiner point of maximal depth in $\mathcal{T}_r$, $v$ is either a terminal (Figure 1.12, left) or a Steiner point with two children $c$ and $d$ that are both terminals (Figure 1.13, left). We now consider each of these two cases in turn.

1. *$v$ is a terminal:* Assume without loss of generality that $au$ is parallel to $sv$, and $bu$ is parallel to $sw$ (Figure 1.12, right). Observe that $v$ and $b$ are on opposite sides of line $\overline{au}$; hence, $u$ and $v$ are on the *same side* of line $\overline{ab}$. In other words, the Steiner point $u$ must be chosen to be on the same side of line $\overline{ab}$ as terminal $v$.



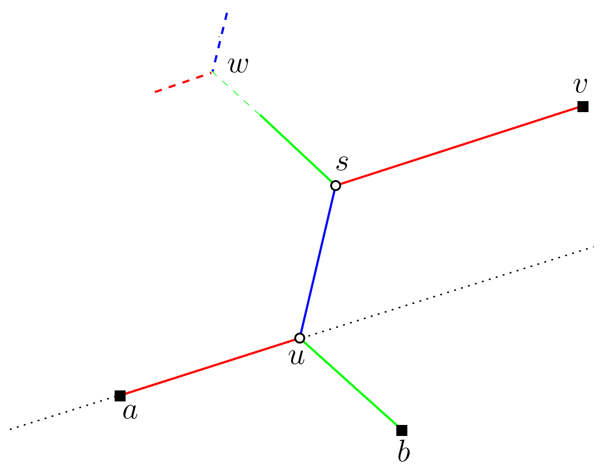
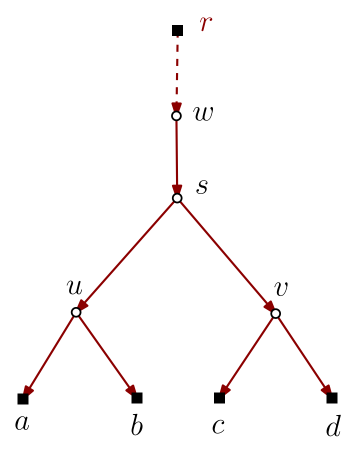


Figure 1.12: Identifying the correct side for a Steiner point where $v$ is a terminal. The rooted topology $\mathcal{T}_r$ is shown on the left, and a possible embedding of part of $T$ is shown on the right. On the right, edges with the same direction are indicated by the same colour.



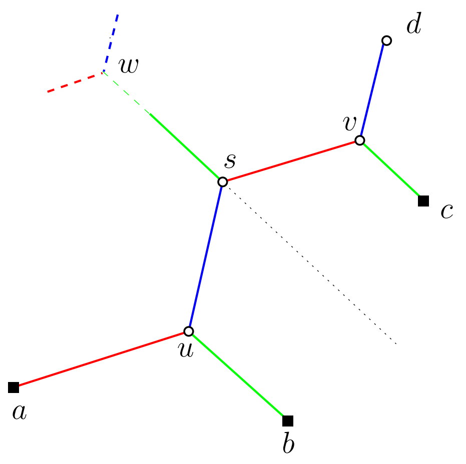


Figure 1.13: Identifying the correct side for a Steiner point (either $u$ or $v$) where $v$ is a Steiner point. The rooted topology $\mathcal{T}_r$ is shown on the left, and a possible embedding of part of $T$ is shown on the right.

2. *$v$ is a Steiner point:* Assume without loss of generality that $au$ is parallel to $sv$ and $bu$ is parallel to $cv$; note that both $bu$ and $cv$ are also parallel to $sw$. Thus, travelling along the path $b \to u \to s \to v \to c$ involves turning through an angle of $\pi/3$ in the same direction (either left or right) at every interior vertex (Figure 1.13, right). Such a path is referred to as a *convex path*. The line $\overline{sw}$ separates the line segments $ab$ and $cd$. Therefore, $ab$ and $cd$ cannot intersect each other: either $c$ and $d$ are both on the same side of line $\overline{ab}$ (in which case the Steiner point $u$ is on the same side of $\overline{ab}$ as $c$ and $d$), or $a$ and $b$ are both on the same side of line $\overline{cd}$ (in which case the Steiner point $v$ is on the same side of $\overline{cd}$ as $a$ and $b$).

Whichever case occurs, we are able to perform a merging operation that correctly identifies a single equilateral point to choose as our next pseudo-terminal. Assuming that the same rooted tree $\mathcal{T}_r$ is used throughout the complete algorithm, identifying the merging operation (and constructing the pseudo-terminal) clearly takes constant time per merging operation. Figures 1.14 and 1.15 illustrate the complete construction on a problem instance with 6 terminals; the merging phase is shown in Figure 1.14 and the reconstruction phase is shown in Figure 1.15.

### Efficient implementation

In the description above, some implementation details of the Melzak-Hwang algorithm were omitted. The complete algorithm is presented as Algorithm 1.1. In the algorithm we assign the topology $\mathcal{T}$ a root at a terminal $r$, and use standard terminology for rooted trees. Also note that the algorithm orders the Steiner points by depth by performing a breadth-first search (BFS) from the root. A BFS algorithm runs in linear time in the number of vertices and edges in a graph [121].

Each of the three main phases of the Melzak-Hwang algorithm — breadth-first search, merging and reconstruction — clearly takes $O(n)$ time, resulting in the following theorem:

**Theorem 1.5** *[212, 280]* ***[Melzak-Hwang Theorem]*** *Given a full Steiner topology $\mathcal{T}$ with $n$ terminals, there is an $O(n)$ time algorithm to either compute a full Steiner tree for $\mathcal{T}$ or decide that no such tree exists.*[9]

Note that the running time is optimal in the sense that the input to the algorithm has space complexity $\Omega(n)$.

### Numerical issues

In the implementation above we have assumed that computers can efficiently represent and perform computations on the coordinates of equilateral points and Steiner points. In

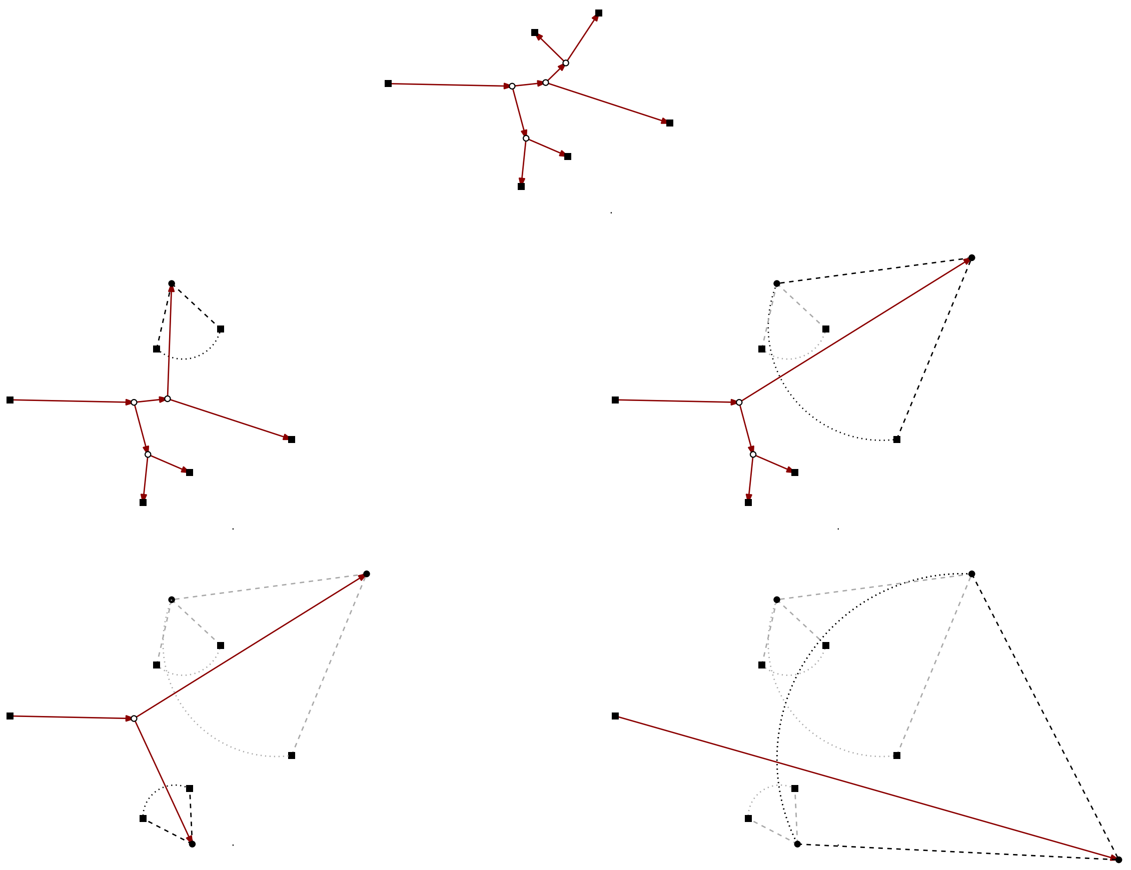


Figure 1.14: Construction of a full Steiner tree for 6 terminals: rooted full Steiner topology (row 1), merging steps 1–4 (left to right on rows 2 and 3).

**Algorithm 1.1:** Melzak-Hwang algorithm

```
Input: Full Steiner topology $\mathcal{T}_r$ with at least 5 terminals, with root at some
       terminal $r$.
Output: Locations of Steiner points in full Steiner tree $T$ for $\mathcal{T}$ (if $T$ exists).

Perform a breadth-first search (BFS) from $r$ in $\mathcal{T}_r$
$S^{\downarrow}$ = Steiner points of $\mathcal{T}_r$ in BFS order
$S^{\uparrow}$ = Steiner points of $\mathcal{T}_r$ in opposite BFS order (deepest first)

// Merging steps
foreach $u \in S^{\uparrow}$ do
    $(a, b)$ = CHILDREN($u$)        // children of vertex $u$ in $\mathcal{T}_r$
    $s$ = PARENT($u$)               // parent of vertex $u$ in $\mathcal{T}_r$
    $v$ = OTHERCHILD($s$,$u$)       // other child of $s$ in $\mathcal{T}_r$ (other than $u$)
    if $v$ is a terminal then
        if $v$ is to the right of $\overrightarrow{ab}$ then
            Locate $u$ at $e_{ab}$ and mark it as a pseudo-terminal
        else
            Locate $u$ at $e_{ba}$ and mark it as a pseudo-terminal
    else
        $(c, d)$ = CHILDREN($v$)
        if line segment $cd$ is to the right of $\overrightarrow{ab}$ then
            Locate $u$ at $e_{ab}$ and mark it as a pseudo-terminal
        else if line segment $cd$ is to the left of $\overrightarrow{ab}$ then
            Locate $u$ at $e_{ba}$ and mark it as a pseudo-terminal
        else if line segment $ab$ is to the right of $\overrightarrow{cd}$ then
            Delete $v$ from $S^{\uparrow}$; Push $u$ back into the front of $S^{\uparrow}$
            Locate $v$ at $e_{cd}$ and mark it as a pseudo-terminal
        else if line segment $ab$ is to the left of $\overrightarrow{cd}$ then
            Delete $v$ from $S^{\uparrow}$; Push $u$ back into the front of $S^{\uparrow}$
            Locate $v$ at $e_{dc}$ and mark it as a pseudo-terminal

// Reconstruction steps
foreach $u \in S^{\downarrow}$ do
    Let $l$ be the line segment from $u$ to PARENT($u$)
    Unmark $u$ as pseudo-terminal
    $(a, b)$ = CHILDREN($u$)
    // Order children such that $a$, $b$, and $u$ appear counter-clockwise on $\triangle abu$
    Set the location of $u$ in $T$ to be the intersection between Steiner arc $\widehat{ab}$ and $l$
    (if the intersection does not exist, then no full Steiner tree exists for $\mathcal{T}_r$)
```

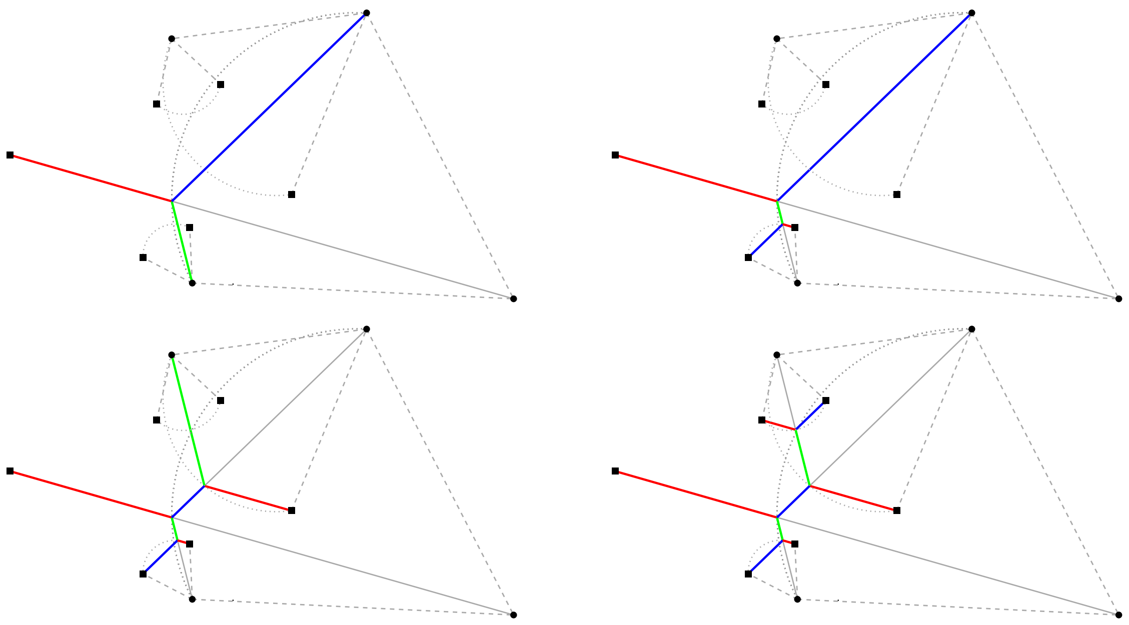


Figure 1.15: Construction of a full Steiner tree for 6 terminals: reconstruction steps 1–4 (left to right on each row), leading to a full Steiner tree (row 2, right)

fact, this is not quite true. Computers can only explicitly handle numbers that have finite representations, such as *integers* or *rational numbers*. Furthermore, constant-time operations can only be performed if the numbers have a *bounded* number of digits. Irrational numbers, such as $\sqrt{3}$ or $\pi$, can be manipulated symbolically, but when an expression involving these symbolic irrational numbers needs to be evaluated, the irrational numbers must be rounded to rational numbers.

Let us assume that the coordinates of the terminals in $N$ are rational numbers. Consider an equilateral point $e_{ab}$ for terminals $a, b \in N$. The point $e_{ab}$ can be obtained by rotating vector $\overrightarrow{ab}$ counter-clockwise by angle $\pi/3$. The corresponding rotation matrix contains the elements $\cos \pi/3$ and $\pm \sin \pi/3$, which are $1/2$ and $\pm\sqrt{3}/2$. Thus, the coordinates of $e_{ab}$ can be written in the form $\alpha + \beta\sqrt{3}$ where $\alpha$ and $\beta$ are rational numbers. In fact, it can be shown that the coordinates of all pseudo-terminals and Steiner points can be written in the form $\alpha + \beta\sqrt{3}$ [169, 354] (Exercise 1.9). In algebra terminology this means that these coordinates belong to the quadratic field $\mathbf{Q}(\sqrt{3})$. Since the length of a full Steiner tree is equal to the distance between a pseudo-terminal and a terminal, we conclude that such a length can be written in the form $\sqrt{\alpha + \beta\sqrt{3}}$, where $\alpha$ and $\beta$ are rational numbers; the same holds for the lengths of the individual edges of the tree.

In the Melzak-Hwang algorithm we can therefore efficiently represent all coordinates and distances symbolically, and can output the coordinates of Steiner points and edge lengths to any precision required (in polynomial time in the number of digits required).

### 1.2.2 Relatively minimal tree for a given full Steiner topology

The Melzak-Hwang algorithm only covers the case where the relatively minimal tree for the given full Steiner topology is *non-degenerate*, that is, where all edges of the relatively minimal tree have non-zero length. For the general problem, where the relatively minimal tree may be degenerate, Hwang and Weng [214] proposed the so-called 'luminary algorithm', which solves the problem for a given full Steiner topology with $n$ terminals in $O(n^2)$ time. The luminary algorithm is a fairly involved generalisation of the Melzak-Hwang algorithm, as significant bookkeeping is needed to handle the various degenerate cases that may appear. (The average running time of the luminary algorithm is $O(n \log n)$, where the average is taken over all full topologies for the set of terminals [417].)

A more practical method for solving the general problem is the *iterative* approach by Smith [354]. In this algorithm, the Steiner points are initially located in the plane using any heuristic or randomised method. Then the Steiner points are iteratively relocated in such a way that the length of the resulting tree is reduced in every iteration. Since the underlying optimisation problem is strictly convex, any iterative method that (strictly) reduces the length of the tree is guaranteed to converge to the unique global minimum [180]. The length function has, unfortunately, non-smooth and complicated behaviour near the optimum, so convergence can be very slow.

The main idea of Smith's algorithm is as follows. Assign labels $\{t_1, \ldots, t_n\}$ to the terminals and labels $\{t_{n+1}, \ldots, t_{2n-2}\}$ to the Steiner points in the topology $\mathcal{T}$. Let $v_i^k$ denote the location of terminal or Steiner point $t_i$ in iteration $k$ (where the location of a terminal, of course, is fixed at its given location throughout the algorithm). In iteration $k+1$, Smith sets up the following vector equation for each Steiner point $t_i \in \{t_{n+1}, \ldots, t_{2n-2}\}$:

$$\sum_{j: t_i t_j \in E(\mathcal{T})} \frac{v_i^{(k+1)} - v_j^{(k+1)}}{|v_i^k - v_j^k|} = 0$$

Note that the equation uses the locations from the previous iteration $k$, and thus the denominator is a constant in iteration $k+1$. Hence, the equation is linear. Each Steiner point converges towards a location where the forces on the Steiner point are in equilibrium; the equation makes the sum of the outgoing unit vectors along the three edges incident to the Steiner point converge to zero. This is only possible when each pair of incident edges meets at an angle of $2\pi/3$ at the Steiner point.

Due to the special tree structure, the system of linear equations can be solved in $O(n)$ time using Gaussian elimination [354]. Furthermore, Smith proves that the algorithm converges with high probability, and that convergence is geometric (and thus fast in practice). Smith's algorithm can be generalised to higher dimensions and other metrics. For dimension 3 or higher, no finite algorithm can exist for the Euclidean problem, so it is necessary to resort to infinite converging processes [354].

## 1.3 Global properties of minimum Steiner trees

In the previous sections we looked at properties of Steiner points, the geometry of full Steiner trees, and the construction of full Steiner trees and relatively minimal trees for a given full Steiner topology. In this section we discuss some important *global* properties of minimum Steiner trees, that is, properties that do not assume that a topology is given.

First we define the related minimum spanning tree problem, and discuss the Steiner ratio — which is the smallest ratio between the length of a minimum Steiner tree and the length of a minimum spanning tree for the same set of terminals. Then we describe a number of global structural properties, that is, necessary properties of minimum Steiner trees; these properties include constraints on the geometric region in which a minimum Steiner tree can appear, and bounds on the length of edges in the tree. Finally, we show that the Euclidean Steiner tree problem is NP-hard, meaning that a polynomial-time algorithm for solving the problem is unlikely to exist.[10]

### 1.3.1 Minimum spanning trees and the Steiner ratio

We begin by defining a shortest network problem that does *not* include Steiner points:

EUCLIDEAN MINIMUM SPANNING TREE PROBLEM IN THE PLANE
**Given**: A set of points $N = \{t_1, \ldots, t_n\}$ lying in the plane.
**Find**: A geometric network $\bar{T} = (V(\bar{T}), E(\bar{T}))$, such that $N = V(\bar{T})$, and such that $|\bar{T}(N)| :+= \sum_{e \in E(\bar{T})} |e|$ is minimised.

A solution to this problem is called a *minimum spanning tree (MST)*. MSTs in edge-weighted graphs can be computed in polynomial time (essentially in linear time in the number of edges [121]). By constructing the complete graph on the set of points $N$, an MST in the plane can be computed in $O(n^2)$ time using, for example, Prim's algorithm for the corresponding graph problem [317]. Prim's algorithm is a classic example of a greedy algorithm.

By exploiting the geometry of the problem, Euclidean MSTs in the plane can be constructed in $O(n \log n)$ time [316]. One well known approach is to use the dual of the Voronoi diagram for $N$, the Delaunay triangulation (which can be constructed in

$O(n \log n)$ time), to identify a subgraph with $O(n)$ edges of the complete graph on $N$ that contains an MST for $N$ (see, for example, [18]).

Clearly, minimum Steiner trees are in general shorter than MSTs for the same terminal set $N$, since minimum Steiner trees are allowed to contain Steiner points. Let $|T(N)|$ and $|\bar{T}(N)|$ denote the length of a minimum Steiner tree and that of an MST, respectively, for $N$. How much shorter can a minimum Steiner tree be relative to an MST for the same set of terminals? Define

$$\rho(N) := \frac{|T(N)|}{|\bar{T}(N)|}$$

to be the ratio between the length of a minimum Steiner tree and an MST for $N$. The *Steiner ratio* $\rho$ is defined as

$$\rho := \inf_{N} \rho(N).$$

That is, the Steiner ratio is the smallest possible ratio between the minimum Steiner tree and the MST lengths for any set of terminals.

**Conjecture 1.6** *[180]* ***[Steiner ratio conjecture]*** *The Steiner ratio for the Euclidean Steiner tree in the plane problem is $\sqrt{3}/2 = 0.866025\ldots$.*

This long-standing conjecture was first given by Gilbert and Pollak [180] in 1968. The conjecture remains open, despite a paper published in 1990 claiming to prove the conjecture (the proof was later shown to be erroneous).[11] The conjectured Steiner ratio is achieved when $N$ is the set of vertices of an equilateral triangle (see Figure 1.1), hence $\sqrt{3}/2$ is an upper bound for $\rho$. It is straightforward to show that $1/2$ is a lower bound for $\rho$ (in any metric); see Exercise 1.10. Currently, the best known lower bound is as follows:

**Lemma 1.7** *[106]* ***[Lower bound for Steiner ratio]*** *The Steiner ratio $\rho$ is bounded below by $\rho_0 = 0.82416874\ldots$, where $\rho_0$ is the unique real root of the polynomial*

$$x^{12} - 4x^{11} - 2x^{10} + 40x^9 - 31x^8 - 72x^7 + 116x^6 + 16x^5 - 151x^4 + 80x^3 + 56x^2 - 64x^2 - 64x + 16$$

*satisfying* $0.8 < \rho_0 < 1$.

Not surprisingly, the proof of this result is highly technical, and is found in [106].

Another approach towards trying to establish the Steiner ratio conjecture is to show that it holds when $n$, the number of terminals, is bounded by some small value. For example, it is easy to show that the conjecture holds when $n \leq 3$; see Exercise 1.11. Rubinstein and Thomas [329], in 1991, showed that the Steiner ratio conjecture holds for $n \leq 6$; in 2009 De Wet [129] extended this result to $n \leq 7$; and more recently, in 2014, Kirszenblat [235] proved it for $n \leq 8$.

As a final remark, we note that if a minimum Steiner tree has terminal-terminal connections, then these may be assumed to come from a single arbitrary MST:

**Lemma 1.8** *[180]* ***[MST edges in minimum Steiner tree]*** *Let $\bar{T}$ be an MST for $N$. Then there exists a minimum Steiner tree $T$ for $N$ such that all terminal-terminal connections (or 2-terminal full Steiner trees) in $T$ are edges from $\bar{T}$.*

**Proof.** Consider some minimum Steiner tree $T$. We show that if $T$ contains a terminal-terminal edge $e$ that is not in $\bar{T}$, then we can replace this edge with an edge $f$ from $\bar{T}$ where $|f| \leq |e|$. From this the lemma follows.

Remove edge $e$ from $T$ and consider the two connected components (subtrees) $T_1$ and $T_2$. Let $t_i$ and $t_j$ be the terminal endpoints of edge $e$. In $\bar{T}$ there is a unique path $P$ from $t_i$ to $t_j$. Let $f$ be the first edge on $P$ that connects a terminal in $T_1$ with a terminal in $T_2$. Edge $f$ reconnects $T_1$ and $T_2$, and clearly $|f| \leq |e|$, since otherwise we could replace $f$ with $e$ in $\bar{T}$ and obtain a shorter spanning tree. ∎

## 1.3.2 Structural properties

In this section we study some necessary geometric conditions that must be satisfied by minimum Steiner trees for a given terminal set $N$, independently of the topology of the tree. These conditions can be checked rapidly, and hence can be used as efficient pruning conditions for eliminating non-feasible topologies, or families of topologies, when attempting to construct a minimum Steiner tree. For example, many of these properties are exploited as part of the GeoSteiner algorithm which we discuss in Section 1.4. The properties fall into two groups: empty regions and edge-length bounds. An *empty region* is a region in the plane that can be shown to be free of Steiner points and/or terminals if certain conditions are fulfilled. An *edge-length bound* provides a lower and/or an upper bound on the length of certain edges in a minimum Steiner tree. We are interested in identifying empty regions and edge-length bounds that can be efficiently computed without having to first compute a minimum Steiner tree.

#### The wedge property and Steiner hulls

We define a *wedge* to be any translation of a convex cone in the plane. The 'wedge property' and its elegant proof were established by Gilbert and Pollak [180].

**Lemma 1.9** ***[The wedge property]*** *Let $W$ be an open wedge having angle $2\pi/3$ or more and containing no elements of $N$. Then $W$ contains no Steiner point of any minimum Steiner tree for $N$.*

**Proof.** Arguing by contradiction, suppose $W$ contains at least one Steiner point $s$. Without loss of generality, we introduce a Cartesian coordinate system $(x, y)$ with positive $x$-axis along the angle bisector of the wedge, and choose $s$ to be a Steiner point in $W$

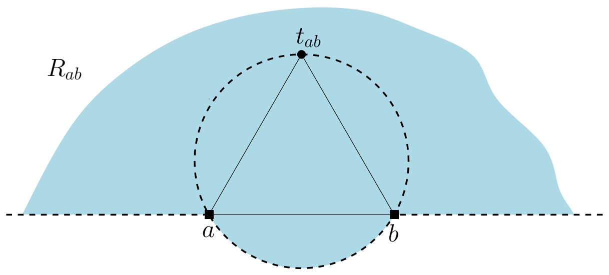


Figure 1.16: One of the two possible regions $R_{ab}$ for two given points $a$ and $b$. The other region is obtained by reflecting the diagram through the line through $a$ and $b$.

with the largest $x$-coordinate. Of the three incident edges at $s$, one leaves $s$ in a direction within $\pm\pi/3$ of the positive $x$-axis. This edge cannot leave $W$ and so cannot end at a terminal. Moreover, its other endpoint has a larger $x$-coordinate than that of $s$, giving a contradiction. ▮

Given two distinct points $a$ and $b$ in the Euclidean plane, let $t_{ab}$ be the third vertex of an equilateral triangle with vertices $a$ and $b$ (hence, $t_{ab}$ equals one of the two equilateral points for $a$ and $b$, $e_{ab}$ or $e_{ba}$). Furthermore, let $C_{ab}$ be the open finite region bounded by the circumcircle of $\triangle abt_{ab}$, and let $R_{ab}$ be the union of $C_{ab}$ and the open half-plane defined by the line through $a$ and $b$ and containing $t_{ab}$, as illustrated in Figure 1.16. Note that $R_{ab}$ is not uniquely defined; there are two possibilities for $t_{ab}$ resulting in two possible choices for the region $R_{ab}$.

This definition leads to the following corollary of the wedge property, which will be used in Section 1.3.3.

**Corollary 1.10** *Let $a$ and $b$ be terminals of a Euclidean minimum Steiner tree $T$. If there exists a region $R_{ab}$, as defined above, containing no terminals of $T$, then that region also contains no Steiner points of $T$.*

**Proof.** This is a simple extension of the wedge property (Lemma 1.9) above. Lemma 1.9 states that any open wedge-shaped region having an angle of $2\pi/3$ and containing no terminals of $T$ also contains no Steiner point of $T$. Region $R_{ab}$ is an infinite union of such wedges, all with $a$ and $b$ on their boundary. ▮

To see the power of this corollary, consider a terminal set $\{a, b, c\}$ such that each angle of the triangle $\triangle abc$ is less than $2\pi/3$. Then the union of the three regions $R_{ab}$, $R_{bc}$ and $R_{ca}$ (where in each case the open circular arc intersects the interior of $\triangle abc$) covers the entire plane with the exception of four points: the three terminals $a, b, c$ and a single point $s$ in the interior of $\triangle abc$ (see Figure 1.3). By Corollary 1.10, $s$ must be the Steiner point of the Euclidean minimum Steiner tree on $\{a, b, c\}$.

**Definition [Steiner hull]**: Given any set of terminals $N$, a *Steiner hull* for $N$ is a bounded region of the plane containing every minimum Steiner tree for $N$.

A Steiner hull can be found by taking the complement of a union of wedges satisfying the wedge property. For example, it follows that the *convex hull* of $N$ is a Steiner hull of $N$, since it is the complement of a union of $\pi$-angled wedges (containing no terminals) bounded by the supporting lines of the convex hull [180].

We now describe a replacement process aimed at reducing the size of a Steiner hull, using terminology suggested by Winter [407].

Let $P_N$ denote a polygon with a subset of the terminal set $N = \{t_1, \ldots, t_n\}$ as its vertices, and such that $P_N$ is the boundary of a Steiner hull for $N$. (We know that at least one such Steiner hull exists, namely the convex hull of $N$.) Let $t_it_j$ be an edge of $P_N$. A replacement of $t_it_j$ by a pair of edges $t_it_k$ and $t_kt_j$, $t_k \in N \setminus \{t_i, t_j\}$, denoted by $t_it_j \to t_it_kt_j$, is said to be *legal* if

- $\triangle t_it_kt_j$ is contained in $P_N$;
- $\triangle t_it_kt_j$ contains no terminals other than its vertices; and
- $\angle t_it_kt_j \geq 2\pi/3$.

Each legal replacement $t_it_j \to t_it_kt_j$ corresponds to a triangle with one of its sides on $P_N$ being replaced by the other two sides and with $t_k$ as a new vertex of the polygon. Note that a legal replacement may result in a non-simple polygon.

**Lemma 1.11** *[111] Let $P_N$ be the boundary of a Steiner hull for $N$. If there is a legal replacement $t_it_j \to t_it_kt_j$ of the edge $t_it_j$ of $P_N$, then the reduced polygon is also the boundary of a Steiner hull for $N$.*

**Proof.** We need to show that no minimum Steiner tree for $N$ enters the interior of $\triangle t_it_kt_j$ or includes the edge $t_it_j$. First, suppose there is at least one Steiner point of some minimum Steiner tree for $N$ in the interior of $\triangle t_it_kt_j$. Then, by the same argument as in the proof of Lemma 1.9, there exists a Steiner point $s$ in the interior of $\triangle t_it_kt_j$ such that some edge incident with $s$ crosses the interior of $t_it_j$, giving a contradiction.

Next, suppose that $\triangle t_it_kt_j$ contains an edge of some minimum Steiner tree $T$ for $N$ that crosses $t_it_j$ in $x_i$ (where, possibly, $x_i = t_i$, but $x_i \neq t_k$), and crosses $t_jt_k$ in $x_j$ (where, possibly, $x_j = t_j$, but $x_j \neq t_k$); see Figure 1.17. Remove $x_ix_j$ from $T$; this splits $T$ into two connected components, one of which contains $t_k$. We can reconnect the components by adding an edge from $t_k$ to either $x_i$ or $x_j$. This results in a tree shorter than $T$ interconnecting $N$, giving a contradiction to minimality. ∎

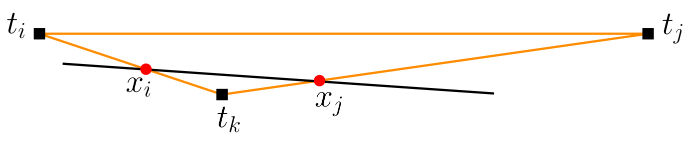


Figure 1.17: An edge crossing the triangle $\triangle t_i t_k t_j$.

This suggests a simple recursive procedure for constructing a minimal area polygonal Steiner hull: beginning with $P_N$ being the boundary of the convex hull of $N$, apply legal replacements to the edges of $P_N$ until no more replacements are possible. This can be done in any order, for example, in depth-first, clockwise order, beginning with a fixed edge of the convex hull. Winter [407] shows that the order in which legal replacements are applied is immaterial; every maximal sequence of replacements results in the same final $P_N$.[12] In addition, Winter also shows that there is a $\Theta(n \log n)$ time and $\Theta(n)$ space algorithm for constructing this minimal Steiner hull, by making use of the Delaunay triangulation of $N$; see [407] for more details.

Related to these above results is the following useful observation, first made by Cockayne [110].

**Corollary 1.12** *If a set of terminals $N$ has the property that all elements of $N$ lie on the boundary $P_N$ of the convex hull of $N$, then every cherry of a minimum Steiner tree $T$ consists of an adjacent pair of vertices of $P_N$.*

This follows immediately from the observations that the convex hull of $N$ is a Steiner hull, and that $T$ has no crossing edges. In fact, by the same argument, a slightly stronger statement is true, namely that the order of terminals in an outer walk of $T$ is the same as the order of terminals around $P_N$.

## The lune property

Another important empty region result is the 'lune property', also established by Gilbert and Pollak [180].

**Definition [Lune]**: Given a line segment $uv$, we define the *lune* of $uv$, $\mathcal{L}(u,v)$, to be the intersection of open discs $D(u)$ and $D(v)$ centred at the points $u$ and $v$, respectively, and each of radius $|uv|$. Equivalently, $\mathcal{L}(u,v)$ is the region consisting of all points $x$ such that $|ux| < |uv|$ and $|vx| < |uv|$.[13]

An example of a lune is shown in Figure 1.18 (left).

**Lemma 1.13** ***[The lune property]*** *If $uv$ is an edge of a minimum Steiner tree $T$, then the lune $\mathcal{L}(u,v)$ does not contain any points of the tree $T$ other than the interior of $uv$.*

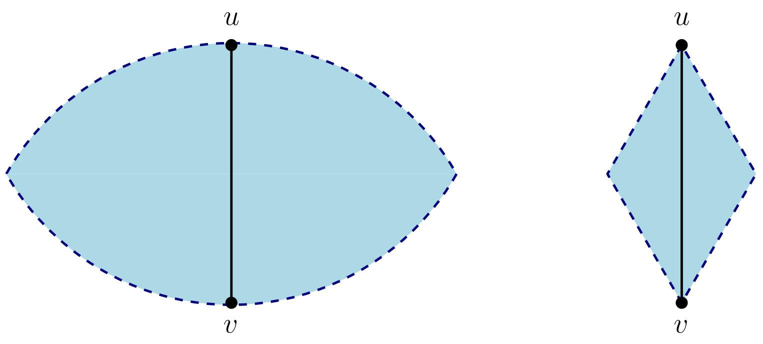


Figure 1.18: The lune (left) and diamond (right) of a line segment $uv$.

The proof of this lemma is straightforward (Exercise 1.13). The lune property forms the basis for one of the most important pruning criteria in the generation phase of the GeoSteiner algorithm. This is described in more detail in Section 1.4.3.

### The diamond property

To conclude the discussion of empty regions, we briefly mention the 'diamond property'. By the *diamond* of a line segment $uv$ we mean the open set of points $x$ such that both angles $\angle xuv$ and $\angle xvu$ are less than $\pi/6$. Hence, the diamond of $uv$ is a rhombus with angles $\pi/3$, $2\pi/3$, $\pi/3$, $2\pi/3$, and with $uv$ as the longest diagonal; see Figure 1.18 (right). Clearly, the diamond of an edge of a minimum Steiner tree is an empty region (apart from its main diagonal) since it is contained in the lune of that edge. Gilbert and Pollak [180] show that if one constructs diamonds for every edge of a minimum Steiner tree $T$, then no two diamonds intersect. The significance of this is that if $n$ is the number of terminals of $T$ and these terminals lie in a region $R$, then the result gives a way of finding an upper bound for $|T|$ in terms of $n$ and the area of a region $U$ which must contain all the diamonds of $T$ and can be computed from $R$. While this upper bound does not appear to have been used much in practice for Euclidean Steiner trees, an equivalent result has been more widely studied for rectilinear Steiner trees, and is discussed in Chapter 3.

### Bottleneck Steiner distance bound

We now turn our attention to edge-length bounds. Assume that $t_i, t_j \in N$ is a pair of distinct terminals. Let $P_{\bar{T}}(t_i, t_j)$ denote the unique path between $t_i$ and $t_j$ in some minimum spanning tree (MST) $\bar{T}$ for $N$.

**Definition [Bottleneck Steiner distance]**: Given two terminals $t_i$ and $t_j$ in $N$, the *bottleneck Steiner distance*, BSD$(t_i, t_j)$, is equal to the length of the longest edge on $P_{\bar{T}}(t_i, t_j)$ for some MST $\bar{T}$.

Note that BSD$(t_i, t_j)$ is well defined, since any MST will result in the same bottleneck Steiner distances; this follows from elementary properties of MSTs. Now, let $P_T(t_i, t_j)$ denote the unique path between $t_i$ and $t_j$ in a minimum Steiner tree $T$. Then we have the following useful lemma:

**Lemma 1.14** ***[Bottleneck Steiner distance bound]*** *Given two terminals $t_i, t_j \in N$, let $P_T(t_i, t_j)$ be a path in some minimum Steiner tree $T$ for $N$. For any edge $e \in P_T(t_i, t_j)$, we have $|e| \leq$ BSD$(t_i, t_j)$.*

**Proof.** Assume that there exists an edge $e \in P_T(t_i, t_j)$ such that $|e| >$ BSD$(t_i, t_j)$. Remove $e$ from $T$, and consider the two connected components (subtrees) $T_1$ and $T_2$. Choose any minimum spanning tree $\bar{T}$ and consider the unique path $p$ between $t_i$ and $t_j$ in $\bar{T}$. The longest edge on this path has length BSD$(t_i, t_j)$. This means that the first edge $f$ on $p$ that connects a terminal in $T_1$ with a terminal in $T_2$ has length $|f| \leq$ BSD$(t_i, t_j)$. By reconnecting $T_1$ and $T_2$ using edge $f$, a shorter tree interconnecting $N$ is obtained — a contradiction. ∎

The necessary condition provided by Lemma 1.14 turns out to be very powerful in practice, and can be supplemented by a generalisation given in Exercise 1.12. Note that the bottleneck Steiner distance bound is independent of the norm; the proof is not reliant on the geometry of the Euclidean plane. In particular, this bound also holds for the Steiner trees discussed in Chapters 2 and 3.

### Four terminal edge-length bounds

There are other edge-length bounds that do exploit Euclidean geometry. We first require a result that provides a useful way of recognising when a full Steiner tree on four terminals is minimum.

**Definition [Acute Steiner tree]**: Let $N$ be a set of four terminals forming the vertices of a convex quadrilateral, and let $o$ be the intersection of the two diagonals of the quadrilateral. Suppose there exists a full Steiner tree $T$ for $N$ where one of the two cherries is, say, $t_1, t_2$. If the angle $\angle t_1 o t_2 \leq \pi/2$, then we say that the Steiner tree $T$ is *acute*.

An example of an acute Steiner tree is illustrated in Figure 1.19.

**Lemma 1.15** *Let $N$ be a set of four terminals forming the vertices of a convex quadrilateral. If an acute full Steiner tree exists for $N$, then it is a minimum Steiner tree. Furthermore, unless the two diagonals of the convex quadrilateral are orthogonal, this is the unique minimum Steiner tree for $N$.*

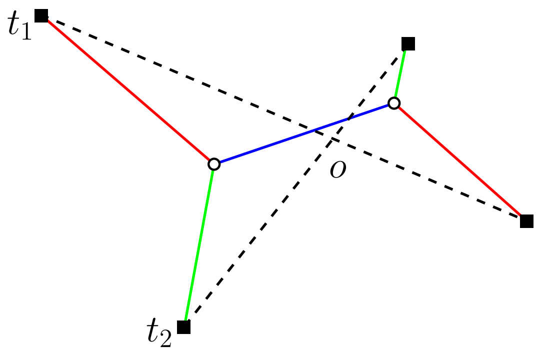


Figure 1.19: A full Steiner tree $T$ on four terminals; $T$ is acute, since $\angle t_1 o t_2 \leq \pi/2$. Hence, by Lemma 1.15, $T$ is a minimum Steiner tree.

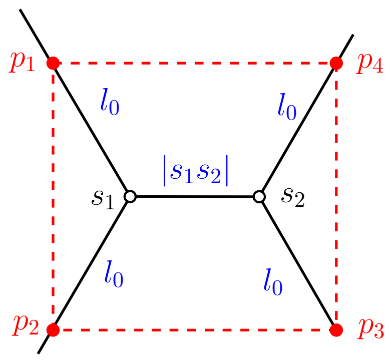


Figure 1.20: The part of $T$ inside the rectangle with corners $p_1$, $p_2$, $p_3$, $p_4$. The lengths of the parts of the edges within this rectangle are indicated in blue.

For details of the proof, see Du et al. [138].[14] The proof relies on showing that the Simpson line for the acute full Steiner tree is shorter than the length of any other Steiner tree on the same set of terminals.

Using the above lemma, we now have the following result.

**Lemma 1.16** *[180] Let $s_1$ and $s_2$ be two adjacent Steiner points in a minimum Steiner tree $T$. Let $l_0$ be the length of the shortest of the four edges (other than $s_1s_2$) incident with $s_1$ or $s_2$. Then $|s_1s_2| \geq (\sqrt{3}-1)l_0$.*

**Proof.** Let $p_1$ and $p_2$ be the two points which lie on the two edges (other than $s_1s_2$) incident with $s_1$ at distance $l_0$ away from $s_1$. Similarly, let $p_3$ and $p_4$ be the two points which lie on the two edges (other than $s_1s_2$) incident with $s_2$ at distance $l_0$ away from $s_2$. See Figure 1.20. The points $p_1$, $p_2$, $p_3$, $p_4$ lie at the corners of a rectangle, and the part of $T$ which interconnects $p_1$, $p_2$, $p_3$, $p_4$ is a full minimum Steiner tree with cherries at $\{p_1, p_2\}$ and $\{p_3, p_4\}$. The side lengths of the rectangle are $|p_1p_2| = |p_3p_4| = \sqrt{3}l_0$ and $|p_1p_4| = |p_2p_3| = l_0 + |s_1s_2|$. Since there exists an acute full Steiner tree for any rectangle, it follows from Lemma 1.15 that $|p_1p_4| \geq |p_1p_2|$, from which the statement of the lemma follows. ∎

There are other similar edge-length bounds that can be proved using Lemma 1.15; see, for example, Exercise 1.14.

### 1.3.3 Computational complexity

In this section we discuss the computational complexity of the Euclidean Steiner tree problem. Discussing the computational complexity demands particular care, as the problem involves irrational numbers (see Section 1.2.1). When a computer outputs the coordinates of Steiner points and/or lengths of edges/trees, it must in general round these irrational numbers to rational numbers. So it is actually more interesting to consider the computational complexity of a *discretised* version of the problem.

First we prove that a certain restricted and discretised version of the Euclidean Steiner tree problem is NP-hard (see Garey and Johnson [171] for a comprehensive introduction to computational complexity). We do this by proving that the decision version of the problem is NP-complete. Based on this result, we are able to show, in Theorem 1.18 below, that the same restricted version of the (original) Euclidean Steiner tree problem is NP-hard — and hence that the general Euclidean Steiner tree problem is NP-hard. The decision version of the restricted problem is the following:

PARALLEL LINES EUCLIDEAN STEINER TREE DECISION PROBLEM
**Instance**: A finite set of points $N$ lying on two parallel lines in the Euclidean plane and a positive integer $L$.
**Question**: Is there a Euclidean Steiner tree $T$ with terminal set $N$ such that the length of $T$ is at most $L$?

As already mentioned, in order to avoid issues related to the theoretical complexity of computing with exact real arithmetic, we in fact consider a *discretised* version of the problem, where distances and coordinates of terminals and Steiner points are constrained to be integers. In the construction below we initially ignore this technical difficulty, and address it at the end of the proof.

The first proof that the discretised Euclidean Steiner tree problem is NP-hard was an intricate and highly technical proof given by Garey, Graham and Johnson [169] in 1977. Our approach here, where the terminals are restricted to parallel lines, is based on an elegant argument by Rubinstein et al. [333], later refined by Brazil et al. [66]. The way we present the construction and proof differs a little from those previous two papers in order to allow us to generalise the result to a much larger class of Steiner tree problems in Chapter 2.

The main idea is to show that the problem can be used to polynomially encode an instance of the subset sum problem, which is well known to be NP-complete [171]:

SUBSET SUM PROBLEM
**Instance**: A set $S = \{d_1, \ldots, d_n\}$ of integers and an integer $d$.
**Question**: Is there a set $J \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in J} d_i = d$?

The main result is as follows. In the proof we let $\mathbf{d}(a, b)$ represent the Euclidean distance between the points or parallel lines $a$ and $b$.

**Theorem 1.17** *The discretised parallel lines Euclidean Steiner tree decision problem is NP-complete.*

**Proof.** Let $S = \{d_1, \ldots, d_n\}$ and $d < \sum_{i=1}^{n} d_i := D$ be a given instance of the subset sum problem. We first show how to use this instance to construct (in polynomial time) an instance of the parallel lines Euclidean Steiner tree problem, and then show that the instance for the subset sum problem is a 'yes' instance if and only if the corresponding instance for the parallel lines Euclidean Steiner tree decision problem is a 'yes' instance. The statement of the theorem then follows.

The construction of the instance for the parallel lines Euclidean Steiner tree problem is as follows. We describe the construction in four stages:

1. Let $V_1, V_1', V_2', V_2$ be four vertical lines ordered from left to right such that

   $$\mathbf{d}(V_1, V_2) \gg \mathbf{d}(V_1, V_1') = \mathbf{d}(V_2', V_2) \gg D. \tag{1.2}$$

   Let $u_0$ be a fixed point on $V_2$, and construct a zigzag path $P$ between $u_0$ and a point on $V_1$ (labelled $v$), such that: $P$ is composed of line segments with alternating polar angles $2\pi/3$ and $\pi/3$; $P$ has $2n$ internal vertices (where $n$ is the cardinality of $S$); and these internal vertices lie alternatively on $V_1'$ and $V_2'$; see Figure 1.21(a).

2. Now from each internal vertex of $P$ on $V_1'$ extend a horizontal line segment to a point on $V_1$. Label these $n$ points $x_1$ to $x_n$ in ascending order. Similarly, from each internal vertex of $P$ on $V_2'$ extend a horizontal line segment to a point on $V_2$, and label these points $u_1$ to $u_n$ in ascending order. Again, this is illustrated in Figure 1.21(a). This results in a Euclidean Steiner tree interconnecting $u_0$, the $u_i$'s, $x_i$'s and $v$ (where in each case $i$ runs from $1$ to $n$). We call this tree the *base tree* $T_x$.

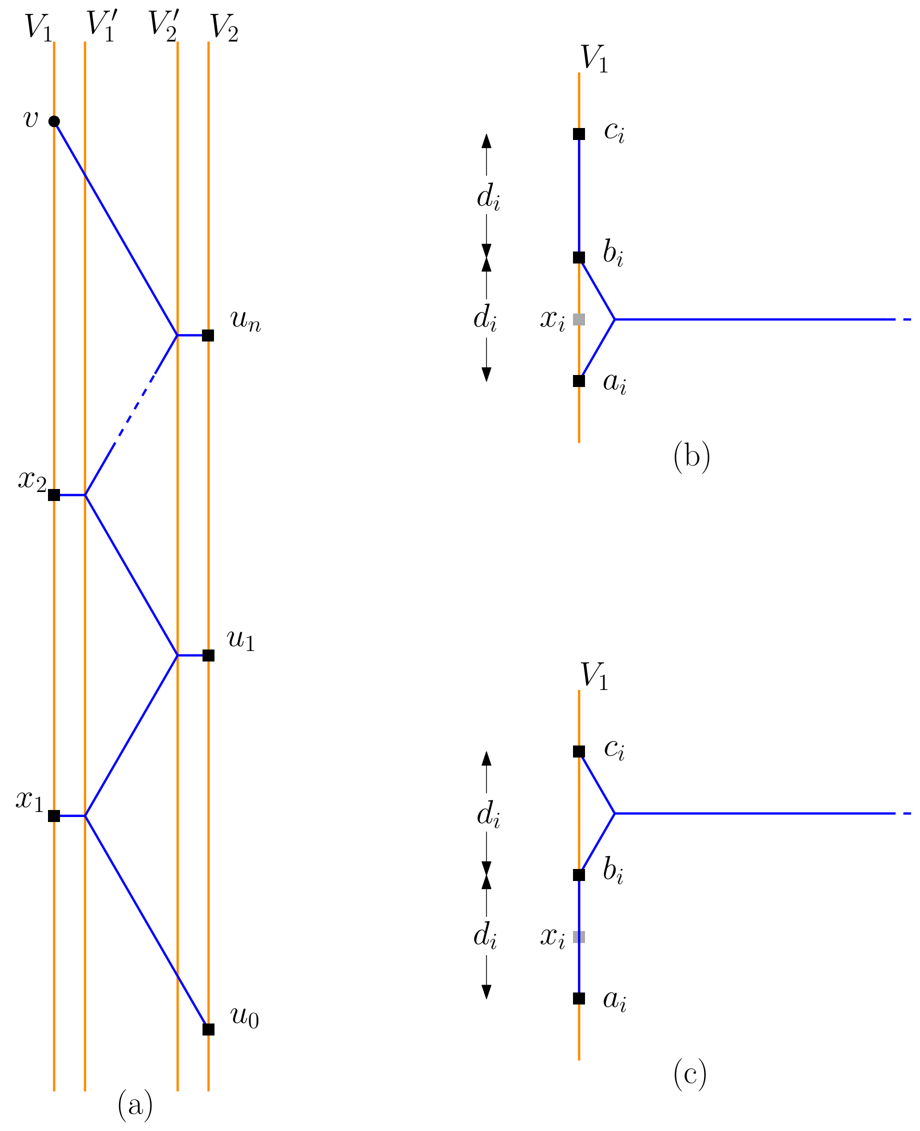


Figure 1.21: Construction of a Euclidean Steiner tree: (a) shows the initial two stages of the construction resulting in the tree shown, the *base tree* $T_x$; (b) shows how $T_v$ connects to each triple $a_i$, $b_i$, $c_i$; and (c) is the alternative connection possible in the tree $T_0$, used in Claim 2.

3. The next stage of the construction is to replace each point $x_i$ by three points on $V_1$ labelled, from bottom to top, $a_i$, $b_i$ and $c_i$, satisfying: $|a_ib_i| = d_i$; $|b_ic_i| = d_i$; and $x_i$ is the midpoint of $a_ib_i$; see Figure 1.21(b).

   We also alter the Steiner tree constructed in Stage 2, so that it connects to $a_i$, $b_i$ and $c_i$, instead of $x_i$. This is done by shortening the horizontal edge by $d_i/(2\sqrt{3})$ on the left and creating a Steiner point at that new left endpoint with two new incident edges with polar angles $2\pi/3$ and $\pi/3$ and each with length $d_i/\sqrt{3}$ connecting to $a_i$ and $b_i$. Finally we connect $b_i$ to $c_i$ with a single (geodesic) edge, i.e., a vertical line segment. This is illustrated in Figure 1.21(b). Let $N_v$ be the set consisting of $u_0$, the $u_i$'s, $a_i$'s, $b_i$'s, $c_i$'s and $v$. We denote the above Euclidean Steiner tree (interconnecting the elements of $N_v$) by $T_v$. We will refer to the topology of the base tree $T_x$ (from Stage 2) as the *base topology* of $T_v$.

Before completing the construction, we establish the following claim:

**Claim 1.** $T_x$ and $T_v$ are each the unique minimum Steiner tree for their respective terminal sets.

**Proof of Claim 1.** Given the differences in scale in Inequality (1.2), consider the limiting case where $\mathbf{d}(V_1, V_1') = \mathbf{d}(V_2', V_2) = 0$. In that case each of $T_x$ and $T_v$ becomes a single zigzag path with polar angles $2\pi/3$ and $\pi/3$ between terminals $\{x_1, \ldots, x_n, v\}$ on $V_1$ and $\{u_0, u_1, \ldots u_n\}$ on $V_2$. The fact that this path is a Euclidean minimum Steiner tree (and hence a minimum Steiner tree) on its vertices follows from Corollary 1.10 by constructing suitable regions: $R_{u_iu_{i+1}}$ for each $i \in \{0, \ldots, n-1\}$; $R_{x_ix_{i+1}}$ for each $i \in \{1, \ldots, n-1\}$; and $R_{x_nv}$ (where in each case the semi-circular open boundary lies between $V_1$ and $V_2$). Taking the union of these regions, it is clear that any Steiner points must coincide with terminals; hence, the minimum Steiner tree coincides with the minimum spanning tree. Furthermore, this minimum spanning tree is easily seen to be unique.

The result now follows immediately by continuity, and the fact that $T_x$ and $T_v$ (in the non-limiting case) are each locally minimal at every Steiner point. ∎

Note that it is straightforward to compute the total Euclidean length of $T_v$ (i.e., $|T_v|$) in terms of $\mathbf{d}(V_1', V_2')$, $\mathbf{d}(V_1, V_1')$, $n$ and the elements of $S$ (see Exercise 1.15). Let $L_v := |T_v|$. Also, we observe that in the main full component of $T_v$, containing all the Steiner points (in other words, all of $T_v$ apart from each of the $(b_i, c_i)$ edges), one of the three orientations of edges is horizontal. We describe such a tree as a *horizontal-edge Steiner tree*.

The final stage of our initial construction is as follows.

4. Let $v_0$ be the point on $V_1$ below $v$ such that $|v_0v| = 2d$. Let $N_0$ be the set $N_v$ where $v$ has been replaced by $v_0$. Let $T_0$ be a minimum Steiner tree for $N_0$.

**Claim 2.** The minimum Steiner tree $T_0$ has the same base topology as $T_v$. Furthermore, for each triple, $a_i$, $b_i$ and $c_i$, the main full component of $T_0$ either connects directly to $a_i$ and $b_i$ only, as in Figure 1.21(b), or to $b_i$ and $c_i$ only, as in Figure 1.21(c).

**Proof of Claim 2.** The first statement follows by the relative scale of the distances involved in Inequality (1.2), using the same argument as in the proof of Claim 1. For the second statement, it is an easy exercise to show that the configurations shown in Figure 1.21(b) and (c) are the only locally minimal ways of connecting the main full component of $T_0$ to $a_i$, $b_i$ and $c_i$. ∎

In other words, we can think of $T_0$ as being the new minimum Steiner tree obtained from $T_v$ by moving the terminal $v$ vertically downwards by $2d$.

**Claim 3.** The following three statements are equivalent:

(A) The answer to the given instance of the subset sum problem is 'yes'.

(B) There exists a horizontal-edge minimum Steiner tree on $N_0$ with the same base topology as $T_v$.

(C) There exists a Steiner tree on $N_0$ with length at most $L_v - \sqrt{3}d$.

**Proof of Claim 3.** The equivalence of the three statements is shown in four steps.

**Step 1: (A) $\Rightarrow$ (B).** Let $T_x$ be the minimum Steiner tree constructed in Stage 2 of the main construction. Suppose we treat $v$ and one of the terminals $x_i$ as 'movable' points, able to move along $V_1$. Then consider the following question: If we move $x_i$ vertically upwards by a distance $\delta$, how does the position of $v$ on $V_1$ change so that $T_x$ remains a horizontal-edge Steiner tree? As Figure 1.22 shows, each horizontal edge incident with a terminal $u_j$ (for $j$ such that $i \leq j \leq n$) increases in length by $2\delta/\sqrt{3}$. In particular, the horizontal edge incident with $u_n$ increases in length by $2\delta/\sqrt{3}$, which implies that $v$ moves downwards by $2\delta$.

We now apply a similar argument to $T_v$. Again, allow $v$ to be a 'movable' point, and consider the effect of changing the connection of the tree at one of the triples $a_i, b_i, c_i$ (from the original connection as shown in Figure 1.21(b) to the alternative connection shown in Figure 1.21(c)) while keeping the tree a horizontal-edge Steiner tree. By the symmetry of the two connection types this is equivalent in its effect on $v$ to moving $x_i$ upwards by $d_i$ in $T_x$; that is, $v$ moves downwards by $2d_i$. This effect is additive across all of the triples, meaning that if we change to the alternative connection scheme at each $i \in J$ where $J \subseteq \{1, \ldots, n\}$ is a set that corresponds to a 'yes' instance of the given subset sum problem, then $v$ moves downwards by $2d$ to $v_0$, giving the required horizontal-edge minimum Steiner tree on $N_0$.

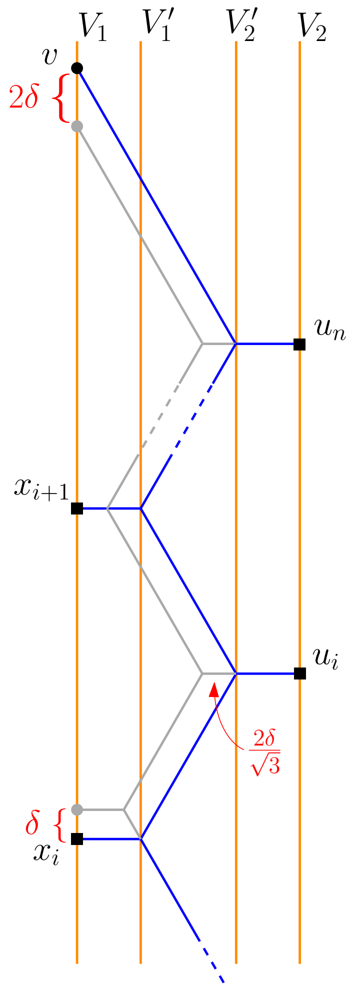


Figure 1.22: Construction for proof of Claim 3 (Step 1).

**Step 2: (B) $\Rightarrow$ (A).** The argument here is similar to that in Step 1. This time we begin with a horizontal-edge minimum Steiner tree on $N_0$ with the same base topology as $T_v$, and treat the terminal $v_0$ as being a 'movable' point on $V_1$. Since $v_0 \neq v$ it follows that there must be at least one $i \in \{1, \dots, n\}$ such that the connection of the tree to $a_i, b_i, c_i$ uses the alternative connection scheme shown in Figure 1.21(c). Let $J' \subseteq \{1, \dots, n\}$ be the set of all such $i$ where this alternative connection scheme is used. If for any $i \in J'$ we change to the original connection scheme (as shown in Figure 1.21(b)) while keeping the tree as a horizontal-edge tree, then, by the same argument as in Step 1, $v_0$ moves upwards by $2d_i$. Now if for every $i \in J'$ we change to the original connection scheme while keeping the tree as a horizontal-edge tree, then it is clear that $v_0$ now coincides with $v$ (since the position of $v_0$ is uniquely determined by the positions of the other terminals, the topology of the tree and the three directions). Since $\mathbf{d}(v_0, v) = 2d$ it follows that

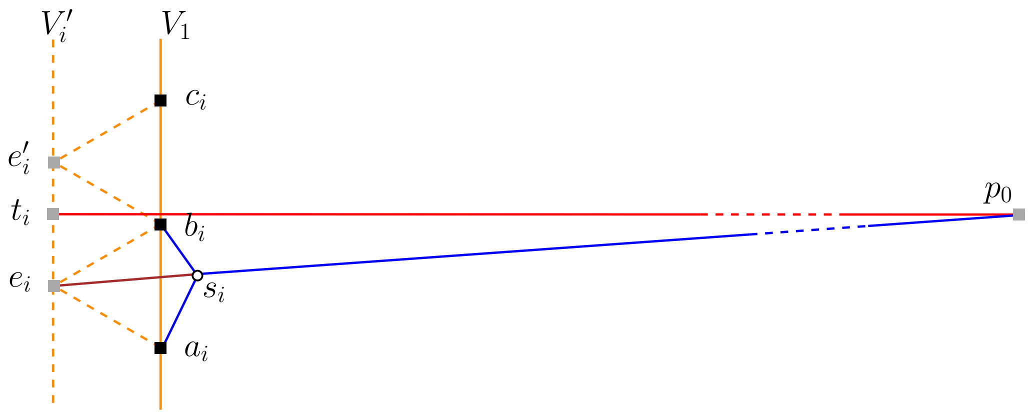


Figure 1.23: Construction for Step 4 of Claim 3.

$\sum_{i \in J'} d_i = d$, and hence $J'$ gives a 'yes' solution to the given instance of the subset sum problem.

**Step 3: (B) $\Rightarrow$ (C).** We first analyse the change in length to $T_x$ under the movement of $x_i$ by $\delta$ described in Step 1 and illustrated in Figure 1.22. For the horizontal edges: the edge incident with $x_i$ decreases in length by $\delta/\sqrt{3}$, each edge incident with $x_j$ for $i+1 \leq j \leq n$ decreases in length by $2\delta/\sqrt{3}$, and each edge incident with $u_j$ for $i \leq j \leq n$ increases in length by $2\delta/\sqrt{3}$. Hence, the total length of the horizontal edges increases by $\delta/\sqrt{3}$. For the main zigzag path: its height decreases by $2\delta$ and hence its length decreases by $4\delta/\sqrt{3}$. Together, these result in an overall decrease in length of $3\delta/\sqrt{3} = \sqrt{3}\delta$ for the whole tree.

It follows for the tree $T_v$ that if we treat $v$ as a 'movable' point, and consider the effect of changing to the alternative connection of the tree at one of the triples $a_i, b_i, c_i$, while keeping the tree a horizontal-edge Steiner tree, the tree decreases in length by $\sqrt{3}d_i$. Hence, by additivity, the horizontal-edge minimum Steiner tree on $N_0$ has length $L_v - \sqrt{3}d$.

**Step 4: $\neg$(B) $\Rightarrow$ $\neg$(C).** To prove this last statement, we argue as follows: choose any set $J' \subseteq \{1, \ldots, n\}$. Let $T_0'$ be the Steiner tree on $N_0$ with the same base topology as $T_v$, where for each $j \in J'$ (and only those $j$) the connection of the tree to $a_j, b_j, c_j$ uses the alternative connection scheme shown in Figure 1.21(c). Now choose some $i \in \{1, \ldots, n\}$. We will assume $i \notin J'$ (but the same argument applies if $i \in J'$). Let $s_i$ be the Steiner point of $T_0'$ adjacent to $a_i$ and $b_i$. Let $e_i = e_{a_i b_i}$ and $e_i' = e_{b_i c_i}$ be the equilateral points (to the left of $V_1$) for $a_i, b_i$ and $b_i, c_i$, respectively; see Figure 1.23. Let $V_i'$ be the vertical line through $e_i$ and $e_i'$. We can use an initial merging step in the Melzak-Hwang algorithm at $s_i$ to replace $a_i$ and $b_i$ by $e_i$, without changing the length or orientations in $T_0'$. We continue to apply the merging process in the Melzak-Hwang algorithm until we have a single Simpson line from $e_i$ passing through $s_i$ to a point $p_0$ such that $|e_i p_0| = |T_0'|$. Again, this is illustrated in Figure 1.23. By the given assumption ($\neg$(B)), this Simpson line is not horizontal. Let $t_i$ be the point on $V_1'$ such that the line segment $t_ip_0$ is horizontal. By the same argument as in Step 3, it follows that $|t_ip_0| = L_v - \sqrt{3}d$. Hence, since $t_ip_0 \perp V_1'$ it follows that $|e_ip_0| = |T_0'| > L_v - \sqrt{3}d$.

**Discretisation and scaling.** Above we have presented a transformation of any instance of the subset sum problem to show that the parallel lines Euclidean Steiner tree decision problem is NP-complete if one ignores arithmetic precision issues. Here we demonstrate that the result remains true when applying a discretisation and scaling that resolves the issues related to computing with irrational numbers.

In the discretised problem, Euclidean distances are rounded up to the nearest integer. Also, it is assumed that terminals and Steiner points can only have integer coordinates. Thus, for a given Steiner tree $T$, performing discretisation increases or decreases the length of every edge by at most 3. Since all trees considered have at most $7n + 1$ edges, every tree is at most length $3 \cdot (7n + 1)$ longer or shorter than before the discretisation.

We need to be able to distinguish between 'yes' and 'no' instances in the discretised problem. More precisely, as shown in the proof of Claim 3 above, we need to be able to distinguish between horizontal-edge minimum Steiner trees and non horizontal-edge minimum Steiner trees.

First we observe that $|e_it_i| \geq 1$. This follows from the arguments in the proof of Claim 3 (Step 1); a positive integer shift $\delta \geq 1$ is needed to turn the tree into an edge-horizontal tree. From this it follows that $|e_ip_0| \geq \sqrt{|t_ip_o|^2 + 1^2} \geq |t_ip_o| + 1/(3|t_ip_o|)$.

Assume that Inequality (1.2) implies that $\mathbf{d}(V_1, V_2) = \Theta(D)$; that is, $\mathbf{d}(V_1, V_2)$ is a (large) constant factor times $D$. Thus $L_v = |T_v| = \Theta(nD)$, and $|t_ip_o| = L_v - \sqrt{3}d \leq \alpha nD$ for some (large) constant factor $\alpha > 0$. Define $\epsilon_n^D := 1/(3\alpha nD)$. We now have that $|e_ip_0| - |t_ip_o| \geq \epsilon_n^D$.

The problem is now scaled by multiplying all terminal coordinates by an integer $K$. One can distinguish between 'yes' and 'no' instances, if $K\epsilon_n^D - 2 \cdot 3 \cdot (7n + 1) \geq 1$. Choosing $K \geq (42n + 7)/\epsilon_n^D = (42n + 7) \cdot 3\alpha nD$ suffices, and results in a polynomial scaling.

It follows that the reduction from the subset sum problem to the discretised problem can be performed in polynomial time, since the number of terminals is linear in $n$, and the coordinates of the terminals can be represented using a number of bits that is polynomial in $n$ and $\log D$. Since the sizes of coordinates of the Steiner points of a solution to the Steiner tree problem are bounded by the coordinates of the terminals, it follows that a certificate can be represented in polynomial space in $n$ and $\log D$, and can be verified within the same time bound. Thus, the discretised parallel lines Euclidean Steiner tree decision problem is in NP. ∎

We have now shown that the *discretised* parallel lines Euclidean Steiner tree decision problem is NP-complete. Consider again the (original) parallel lines Euclidean Steiner tree decision problem. We do not know if this problem is NP. The technical difficulty is that no polynomial algorithm currently exists for deciding if $\sum_{i=1}^{k} \sqrt{L_i} \leq L$ for some given set of integers $L_1, \ldots, L_k$ and $L$; the size of the input is measured here as the total number of bits used to represent the given integers.

Even if the problem may not be in NP, we can argue that it is at least as hard as any problem in NP (and thus NP-hard) [169]. For assume that there exists a polynomial-time algorithm for the parallel lines Euclidean Steiner tree problem — where the irrational numbers in the output are represented symbolically. As argued in Section 1.2.1, such a symbolic representation enables us to obtain an output with any required precision (in polynomial time in the number of digits required). This means that the discretised parallel lines Euclidean Steiner tree decision problem — and hence all NP-complete problems — can be solved in polynomial time.

**Theorem 1.18** *The parallel lines Euclidean Steiner tree problem is NP-hard — and hence the general Euclidean Steiner tree decision problem is NP-hard.*

## 1.4 GeoSteiner algorithm

The fact that the Euclidean Steiner tree problem is NP-hard does not preclude the possibility of developing efficient exact algorithms for solving real-world problem instances. NP-hardness is a measure of worst-case performance, and such difficult instances may almost never occur in practice.

The *GeoSteiner* algorithm is by far the most efficient exact algorithm for computing a minimum Steiner tree. The running time of GeoSteiner shows good average behaviour and scaling in practice. The GeoSteiner approach was originally proposed by Winter [404] in 1985, but some of the algorithmic ideas have their origin in the 1960s and 1970s.[15] More recently, significant improvements to GeoSteiner have been implemented, allowing the computation of minimum Steiner trees with several thousand terminals, even for a variety of distance metrics [12, 190, 297, 388, 389, 390, 391, 409, 429].

In this section we first present a top-level description of the GeoSteiner algorithm. We next discuss the full Steiner tree generation algorithm, first presenting the basic concepts and then giving some key details, particularly for pruning methods. It should be noted that the GeoSteiner algorithm is, in a sense, a framework that can be filled out in various ways. In this book we pay particular attention to the most important parts of the algorithm — from a performance point of view.

### 1.4.1 Top-level algorithm

A naïve algorithm for computing a minimum Steiner tree is to enumerate all full Steiner topologies (see Section 1.1.3) for every subset of terminals, and then apply the Melzak-Hwang algorithm (see Section 1.2.1) to compute a full Steiner tree (FST) for each such full Steiner topology. A minimum Steiner tree is then obtained by identifying a subset among the constructed FSTs that interconnects $N$ and has minimum length. However, since the number of terminal subsets is exponential in $N$ — and the number of full Steiner topologies for each subset is super-exponential — this algorithm would exhibit very bad scaling. It would be infeasible in practice to compute minimum Steiner trees with more than, say, 15 terminals using this naïve algorithm.

It turns out that a significant amount of the work in the naïve algorithm is wasted. Firstly, most of the full Steiner topologies have no associated FST. Secondly, most of the constructed FSTs are not shortest networks for their terminals and do not fulfil the structural properties presented in Section 1.3.2.

GeoSteiner follows the same two-phase approach as the naïve algorithm, but reduces the work significantly by implicit instead of explicit enumeration of FSTs (for all subsets and all full Steiner topologies). FSTs that do not fulfil necessary structural properties are eliminated early — and in most cases without direct geometric construction. The set of generated FSTs should be *sufficient* in the sense that it must contain the full components of at least one minimum Steiner tree; furthermore, the generation of FSTs should be *efficient* in the sense that as many of the FSTs as possible that cannot be part of some minimum Steiner tree should be removed — or, ideally, never considered at all.

The FST generation phase of GeoSteiner is followed by the FST concatenation phase, where subsets of generated FSTs are combined to form a minimum Steiner tree. The overall GeoSteiner algorithm is given in Algorithm 1.2. The details of the algorithm will be given in the following sections. Some parts of the generation phase are heavily metric-dependent, and some parts can be adapted to other metrics as shown in the following chapters of the book. The concatenation phase is completely metric-independent, so an algorithm for solving this problem can be directly applied to other metrics.

### 1.4.2 Enumeration of equilateral points, branches and branch trees

In this section we introduce some of the core concepts of the FST generation phase of GeoSteiner. Consider the three FSTs in Figure 1.24, each indicated by a different colour. The terminal pair $\{a, b\}$ forms a cherry in each of these FSTs, and the Steiner points are located on a common Steiner arc $\widehat{ab}$. Not only do the three FSTs share a sub-topology (namely the cherry $\{a, b\}$), but the corresponding Steiner points are also on the same side of the line $\overline{ab}$.

**Algorithm 1.2:** GeoSteiner algorithm

```
Input: Set of points (terminals) $N$ in the plane.
Output: A minimum Steiner tree for $N$.
1
2 // Generation phase
3 Construct a sufficient set of full Steiner trees $\mathcal{F} = \{T_1, T_2, \ldots, T_m\}$ by efficient
  enumeration
4
5 // Concatenation phase
6 Identify a subset $\mathcal{F}^* \subseteq \mathcal{F}$ such that $\mathcal{F}^*$ interconnects $N$ and has minimum total
  length
```

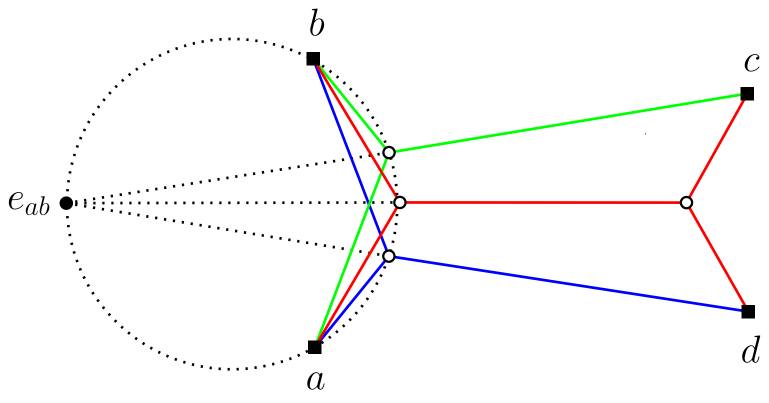


Figure 1.24: Three FSTs sharing an equilateral point and having a Steiner point on the Steiner arc $\widehat{ab}$. The green FST spans $\{a, b, c\}$, the blue FST spans $\{a, b, d\}$ and the red FST spans $\{a, b, c, d\}$.

Recall the definition of *equilateral points* from Section 1.1.1. Instead of enumerating FSTs directly, GeoSteiner enumerates equilateral points with associated subtopologies as described below. Equilateral points fix a part of the geometry of the associated FSTs, which makes it possible to check some of the necessary structural properties early in the enumeration algorithm. For example, imagine that there existed a fifth terminal $t$ halfway between $a$ and $b$ in Figure 1.24, as shown in Figure 1.25. Now consider a Steiner point $s$ on Steiner arc $\widehat{ab}$. No matter where $s$ is located on $\widehat{ab}$, terminal $t$ would be inside one of the two lunes defined by edges $as$ and $bs$. (For more on areas covered by lunes, see [180].) This implies that the equilateral point $e_{ab}$ has no feasible Steiner point on $\widehat{ab}$ and can be pruned. Hence, none of the three FSTs shown in Figure 1.25 would be generated.

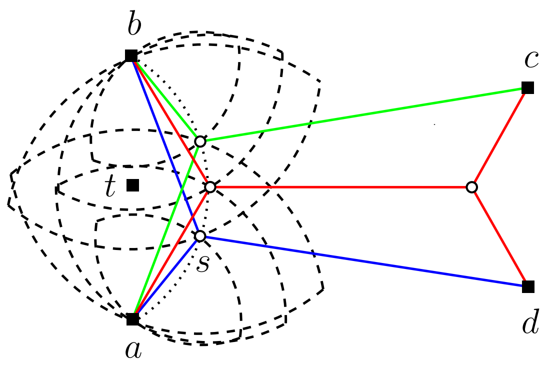


Figure 1.25: Lunes for edges $as$ and $bs$ in FSTs.

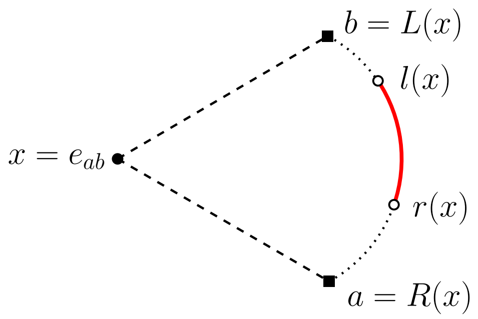


Figure 1.26: An equilateral point $x = e_{ab}$ for $a$ and $b$ and a feasible subarc (indicated in red) for the associated Steiner point.

## Equilateral points and feasible subarcs

The idea of the FST generation algorithm is to enumerate equilateral points bottom-up starting from the terminals. The algorithm simulates the Melzak-Hwang algorithm but attempts to construct FSTs for all possible full Steiner topologies in parallel.

Consider an equilateral point $x = e_{ab}$. Define $R(x) = a$ and $L(x) = b$ to be the *base points* of $x$ (Figure 1.26). Note that $R(x)$ and $L(x)$ may be terminals or equilateral points. The equilateral point $x$ forms the root of a binary tree in which the children of an equilateral point are its base points — and where terminals are leaves. The *order* of $x$ is the depth of the associated binary tree. Hence, if $R(x)$ and $L(x)$ are terminals, $x$ is of order one; for convenience, terminals are considered to be equilateral points of order zero.

The Steiner point for a given equilateral point $x$ must be located somewhere on the Steiner arc $\widehat{R(x)L(x)}$ of $x$ (Figure 1.26). Usually only a part of the Steiner arc is in fact feasible. Therefore, GeoSteiner maintains a *feasible subarc* $\widehat{r(x)l(x)}$ for $x$: a feasible Steiner point can only be located on this subarc (and not including its endpoints). Initially we have $r(x) = R(x)$ and $l(x) = L(x)$, and by employing a number of pruning tests, the subarc is iteratively reduced — and possibly eliminated completely, in which case the equilateral point $x$ can be removed altogether.

### Branches and branch trees

On a slightly more abstract and metric-independent level, we may consider the FST generation of GeoSteiner as an enumeration of so-called *branches*. Given any FST $T$ and an edge $pq$ in $T$, imagine that we cut edge $pq$ at its midpoint. We obtain two *branch trees*: one rooted at $p$ having an 'unfinished' edge leaving $p$ along $pq$, and another rooted at $q$ having an 'unfinished' edge leaving $q$ along $qp$. More formally, branch trees and branches are defined as follows.

**Definitions [Branch trees, branches]**: A *branch tree* $B$ is a tree spanning a set of terminals $N_B$, and containing a ray (known as the *stem*) emanating from one of the vertices $v$ of the tree (known as the *root*), such that for any point $a(\neq v)$ on the stem the union of $av$ with all edges of $B$ other than the stem is an FST for $\{a\} \cup N_B$. A *branch* $\mathcal{B}$ is a (possibly infinite) set of branch trees, each of which spans a common set of terminals $N(\mathcal{B})$, such that every branch tree $B \in \mathcal{B}$ has the same topology.

Intuitively, we can think of a branch tree as an FST spanning a set of terminals and a single point at infinity. Note that for the Euclidean metric, equilateral points and the associated binary trees are in fact branches: for a given equilateral point/branch, each Steiner point on the feasible subarc forms the root of a branch tree. We define the *size* of a branch $\mathcal{B}$ (respectively branch tree $B$) to be the number of terminals $|N(\mathcal{B})|$ spanned by $\mathcal{B}$ (respectively $B$). A branch tree of size 1 consists of a single terminal with a single emanating stem; such a branch tree has no Steiner points, and the terminal is the root of the branch tree. A branch tree of size $k \geq 2$ has $k$ terminals and $k-1$ Steiner points.

The generation phase of GeoSteiner enumerates branches of increasing size — essentially using a dynamic programming approach. For a set of $n$ terminals, there are $n$ branches of size 1, each corresponding to a terminal $t_i \in N$ with an infinite set of stems pointing in all possible directions. Branches of size $k$ are obtained by joining branches of size $l$ with branches of size $k-l$, where $l$ runs from 1 to $\lfloor k/2 \rfloor$. More specifically, the new set of branch trees is obtained by intersecting the stems from the branch trees of the branches of size $l$ and $k-l$. The meeting angle of the intersection (which is a new Steiner point and root) must be $2\pi/3$, which uniquely determines the new stem of the combined branch tree.

An FST is obtained by identifying two branches where the stems can be made to overlap with each other in opposite directions. The complete FST generation algorithm

**Algorithm 1.3:** GeoSteiner FST generation algorithm

**Input**: Set of points (terminals) $N$ in the plane.
**Output**: A set of FSTs $\mathcal{F} = \{T_1, T_2, \ldots, T_m\}$ that is guaranteed to contain the full components of at least one minimum Steiner tree for $N$.

```
1
2  Let $\mathcal{F}$ be the $n-1$ edges of any minimum spanning tree for $N$
3  Let $\Gamma_1$ be the set of branches of size 1 (set of terminals $N$)
4
5  for $k = 2$ to $n$ do
6      // Generate $\Gamma_k$, the set of branches of size $k$, and generate FSTs of size $k$
7      Let $\Gamma_k = \emptyset$
8      for $l = 1$ to $\lfloor k/2 \rfloor$ do
9          foreach $\mathcal{B}_1 \in \Gamma_l$ do
10             foreach $\mathcal{B}_2 \in \Gamma_{k-l}$ do
11                 // Generate a new branch
12                 if $k < n$ then
13                     Construct a new branch $\mathcal{B}$ from $\mathcal{B}_1$ and $\mathcal{B}_2$
14                     if $\mathcal{B}$ is feasible and passes pruning tests then
15                         Add $\mathcal{B}$ to $\Gamma_k$
16                 // Generate a new FST
17                 if $k > 2$ then
18                     Construct a new FST $T$ from $\mathcal{B}_1$ and $\mathcal{B}_2$
19                     if $T$ is feasible and passes pruning tests then
20                         Add $T$ to $\mathcal{F}$
```

is given as Algorithm 1.3. The presented algorithm should be seen as a template that can be filled out in a number of different ways — even for different metrics. Note that from Lemma 1.8 we know that the only FSTs spanning 2 terminals that need to be included in $\mathcal{F}$ are the $n-1$ edges from any minimum spanning tree for $N$ (line 2 in Algorithm 1.3).

## 1.4.3 Pruning of equilateral points/branches and full Steiner trees

In this section we describe the most important tests for pruning equilateral points/branches (line 14 in Algorithm 1.3) and for pruning FSTs (line 19 in Algorithm 1.3). Pruning tests for equilateral points/branches are based on reducing the feasible subarc of equilateral points [404, 409]. We begin with simple projection tests that basically use the fact that edges meet at angles of $2\pi/3$ at Steiner points. We then describe tests based on the lune

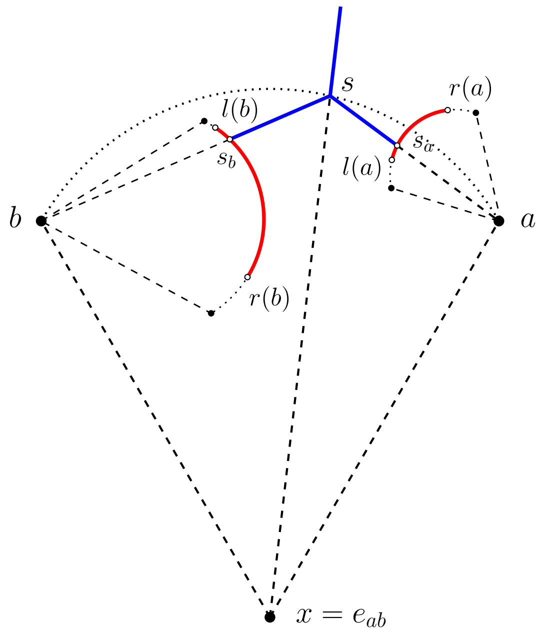


Figure 1.27: Construction of Steiner point $s$ for equilateral point $x = e_{ab}$.

property and the bottleneck Steiner distance bound (see Section 1.3.2). Finally, we present some so-called upper bound tests that use the fact that we can discard branch trees that cannot be used to construct a minimum FST on a subset of the terminals. Throughout this section we assume that the coordinates of the terminals are given as rational numbers.

## Projections

Consider an equilateral point $x = e_{ab}$ where the base points $a$ and $b$ are of non-zero order. A necessary condition for the existence of $x$ is that it is possible to construct at least one FST that has a Steiner point on the Steiner arc $\widehat{ab}$ (and the topology given by the construction of $x$). More specifically, there must exist a Steiner point $s$ on $\widehat{ab}$ such that the ray from $a$ through $s$ intersects the feasible subarc $\widehat{r(a)l(a)}$ in a point $s_a$ strictly between $a$ and $s$; similarly, there must exist a ray from $b$ through $s$ that intersects the feasible subarc $\widehat{r(b)l(b)}$ in a point $s_b$ strictly between $b$ and $s$ (Figure 1.27). Note that $s_a s$ and $s_b s$ are possible edges in an FST.

The Steiner point $s$ can be viewed as a *projection* of $s_a$ (or $s_b$) onto Steiner arc $\widehat{ab}$. A necessary condition for the existence of an equilateral point $x$ is therefore that there exists a projection of some point on the feasible subarc $\widehat{r(a)l(a)}$ onto $\widehat{ab}$; similarly for the feasible subarc $\widehat{r(b)l(b)}$. Below we exactly identify the feasible projections for the equilateral point $a$; the feasible projections corresponding to $b$ are identified in a similar

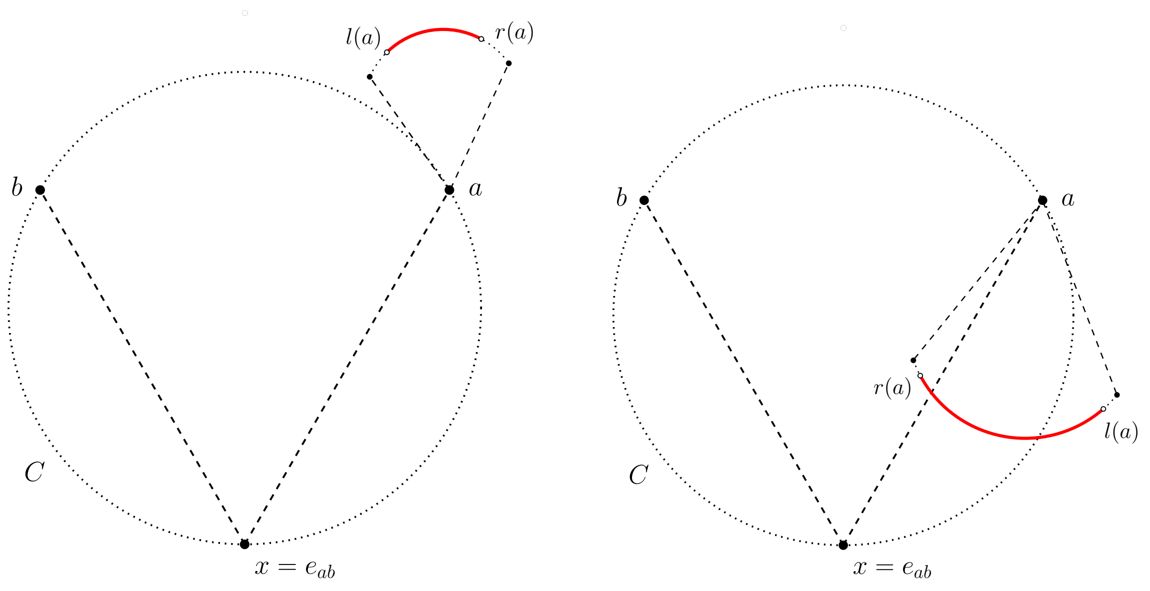


Figure 1.28: Case 1: $r(a)$ and $l(a)$ are outside $C$ (left). Case 2: $r(a)$ is inside $C$, $l(a)$ is outside $C$ (right). In both cases $x$ can be pruned.

manner. The final set of feasible projections is the intersection of these two projection sets. If this intersection set is empty, the equilateral point $x$ can be pruned.

Let $C$ be the circle circumscribing $a$, $b$ and $x$. We now consider the following cases of the relative locations of $r(a)$ and $l(a)$ with respect to circle $C$ (illustrated in Figures 1.28 to 1.30):

- *Case 1.* $r(a)$ and $l(a)$ are outside $C$. The equilateral point $x$ can be pruned, since no feasible projection onto $\widehat{ab}$ exists (Figure 1.28, left).

- *Case 2.* $r(a)$ is inside $C$, $l(a)$ is outside $C$. The projection of $r(a)$ onto $C$ is not on $\widehat{ab}$, so no feasible subarc exists. The equilateral point $x$ can be pruned (Figure 1.28, right).

- *Case 3.* $r(a)$ is outside $C$, $l(a)$ is inside $C$. Let $p$ be the projection of $l(a)$ onto $C$, and let $q$ be the intersection of $\widehat{r(a)l(a)}$ with $\widehat{ab}$ (if $q$ does not exist, $x$ can be pruned). If $p$ is on $\widehat{ab}$, then $\widehat{pq}$ forms a feasible subarc for $x$; otherwise, $\widehat{bq}$ forms a feasible subarc for $x$ (Figure 1.29).

- *Case 4.* Both $r(a)$ and $l(a)$ are inside $C$. This implies that $\widehat{r(a)l(a)}$ is also completely inside $C$. Let $p$ be the projection of $l(a)$ onto $C$, and let $q$ be the projection of $r(a)$ onto $C$. If both $p$ and $q$ are on $\widehat{ab}$, then $\widehat{pq}$ forms a feasible subarc for $x$.

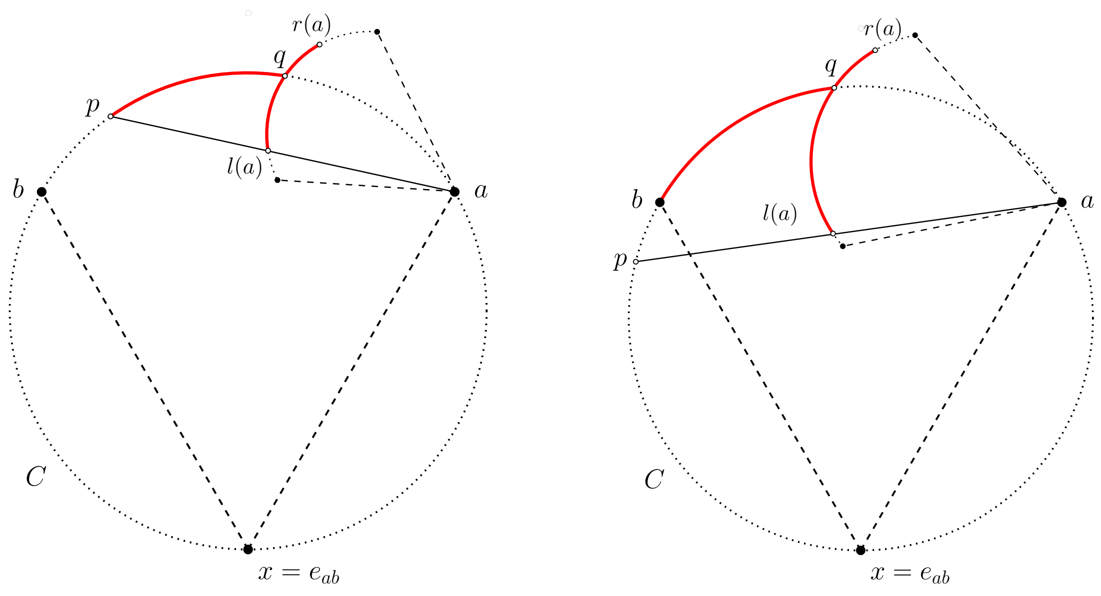


Figure 1.29: Case 3: $r(a)$ is outside $C$, $l(a)$ is inside $C$.

If only $q$ is on $\widehat{ab}$, then $\widehat{bq}$ forms a feasible subarc for $x$ (Figure 1.30). In all other cases, $x$ can be pruned.

For a given equilateral point, the projection test takes constant time; furthermore, it can be performed using simple geometric constructions that do not involve trigonometric functions [391, 409].

## The lune property

Recall that a lune $\mathcal{L}(u, v)$ is defined as the set of points that are strictly within distance $|uv|$ of both $u$ and $v$. If $uv$ is an edge in a minimum Steiner tree, then $\mathcal{L}(u, v)$ cannot contain any terminal (Lemma 1.13). This property can be used to design a very effective pruning test as follows.

Assume that a feasible subarc $\widehat{l(x)r(x)}$ for the equilateral point $x = e_{ab}$ has been identified using the projection test. If $a$ is of non-zero order, let $u$ be the point on $\widehat{l(a)r(a)}$ that is projected onto point $v = l(x)$ (Figure 1.31, left); if $a$ is of order zero (i.e., a terminal), let $u = a$. Assume that there exists a terminal $t$ inside $\mathcal{L}(u, v)$. We may now push $v = l(x)$ toward $r(x)$ on $\widehat{ab}$ until the corresponding lune does not contain $t$ (Figure 1.31, right). If $v$ is pushed all the way to $r(x)$, equilateral point $x$ can be pruned.

The test can be repeated for endpoint $r(x)$, where $r(x)$ is pushed toward $l(x)$. Furthermore, the test is repeated until the lunes corresponding to $l(x)$ and $r(x)$ are free of

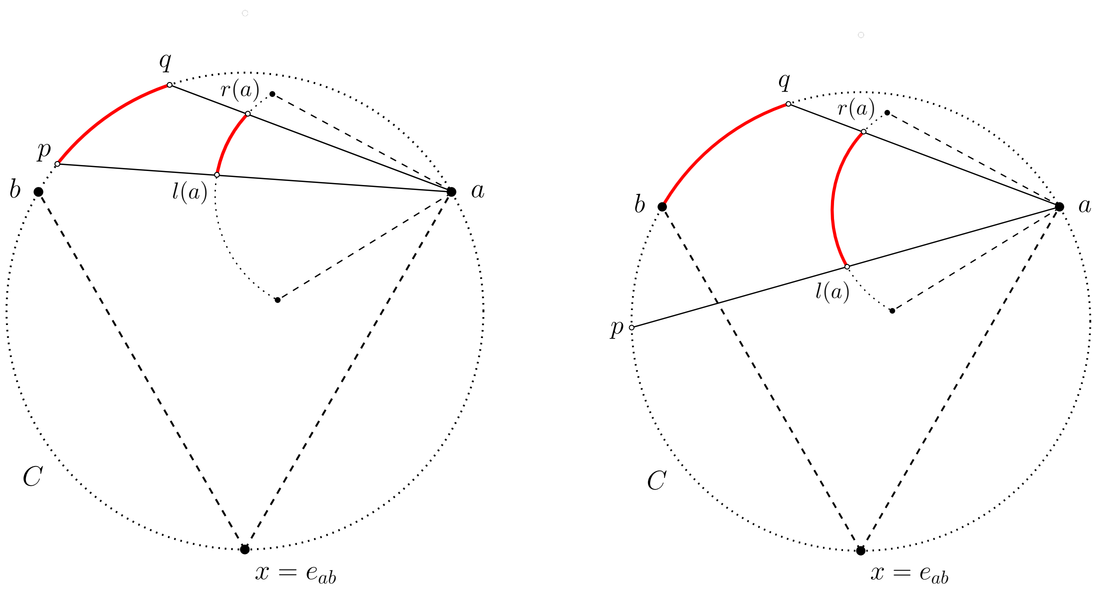


Figure 1.30: Case 4: Both $r(a)$ and $l(a)$ are inside $C$.

terminals — or the feasible subarc is empty and equilateral point $x$ can be pruned.

Similarly to the projection test, the lune property test can be performed using elementary geometric constructions and computations that do not involve trigonometric functions [391, 409]. The test takes constant time for each push, plus the time needed to identify a terminal inside the corresponding lune (if any).

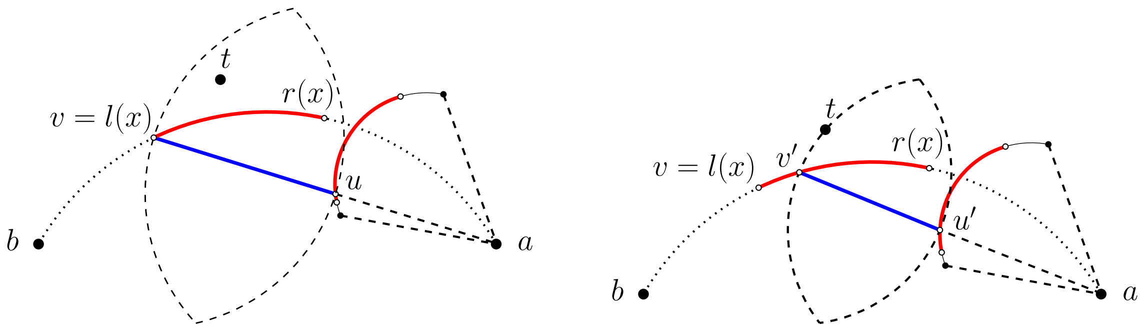


Figure 1.31: Lune property test. Terminal $t$ is inside lune $\mathcal{L}(u, v)$ (left). Endpoint $v = l(x)$ is pushed toward $r(x)$ on $\widehat{ab}$ to a point $v'$ such that $t$ is no longer in $\mathcal{L}(u', v')$ (right).

### The bottleneck Steiner distance bound

Consider the equilateral point $x = e_{ab}$, where each of the points $a$ and $b$ is either a terminal or equilateral point; let $N(a)$ and $N(b)$ be the terminals that are involved in the construction of $a$ and $b$, respectively. A Steiner point $s$ on $\widehat{ab}$ is the root of a branch tree $B_s$ with two children (also branch trees) that interconnect the terminals in $N(a)$ and $N(b)$, respectively. Any path in $B_s$ between a terminal $t_a \in N(a)$ and $t_b \in N(b)$ includes the Steiner point $s$, and, more specifically, includes the adjacent edges $s_a s$ and $s_b s$ (Figure 1.27).

The bottleneck Steiner distance $\text{BSD}(t_a, t_b)$ bounds the length of each edge on a Steiner tree path between $t_a$ and $t_b$ (Lemma 1.14). Let

$$B = \min_{t_a \in N(a),\ t_b \in N(b)} \text{BSD}(t_a, t_b)$$

be the minimum pairwise bottleneck Steiner distance between a terminal in $N(a)$ and a terminal in $N(b)$. Then we must have $|s_a s| \leq B$ and $|s_b s| \leq B$.

This edge-length bound results in a very powerful pruning test, that can be implemented in a similar manner as the lune property test. Consider Figure 1.31 (left). If we have $|uv| > B$, then we may push $v = l(x)$ toward $r(x)$ on $\widehat{ab}$ until $|uv| = B$, or until $l(x)$ is pushed all the way to $r(x)$ and equilateral point $x$ can be pruned. As in the previous tests, the bottleneck Steiner distance test can be performed using elementary geometric constructions and computations that do not involve trigonometric functions [391, 409]. The test takes constant time, plus the time needed to compute $B$ (which depends on the data structure used for holding pairwise bottleneck Steiner distances, as discussed below).

### Upper bounds

Let $s$ be a Steiner point on the feasible subarc $\widehat{l(x)r(x)}$ for the equilateral point $x = e_{ab}$. We define the *length* of the branch tree $B_s$ rooted at $s$ to be the sum of the lengths of all edges of $B_s$ other than the unbounded edge; hence, the length of $B_s$ is $|sx|$, as $sx$ is part of a Simpson line (see Section 1.1.1). Since it is assumed that $B_s$ will merge with other branch trees to form a minimum Steiner tree, it follows that $B_s$ must have minimum length; that is, it must be a minimum length interconnection of $N(x) \cup \{s\}$. The challenge is that $s$ can appear anywhere on the feasible subarc $\widehat{l(x)r(x)}$, so $s$ is a 'floating terminal'. However, it is not difficult to see that $|sx|$ is a strictly concave function as $s$ moves from $l(x)$ to $r(x)$ on $\widehat{l(x)r(x)}$, so the minimum is attained at one of the endpoints. Let $LB = \min\{|l(x)x|, |r(x)x|\}$ be that minimum — or lower bound on the length of any branch tree for equilateral point $x$.

A number of different heuristics can now be applied to provide an upper bound on the length of a Steiner tree that interconnects $N(x)$ and some point on $\widehat{l(x)r(x)}$. Let

$UB(a)$ and $UB(b)$ be upper bounds on the length of Steiner trees that interconnect the terminals $N(a)$ and $N(b)$, respectively. (These bounds can be computed and stored when equilateral points $a$ and $b$ are generated.) Let $s_m$ be the midpoint of $\widehat{l(x)r(x)}$. Now the following is a legal upper bound:

$$UB = UB(a) + \min_{t_a \in N(a)} |t_a r(x)| + |r(x)s_m| + UB(b) + \min_{t_b \in N(b)} |t_b l(x)| + |l(x)s_m|.$$

Note that replacing $s_m$ in the above equation by any other point selected from $\widehat{l(x)r(x)}$ results in a lower value on the right-hand side of the equation.

The equilateral point $x$ can now be pruned if $UB < LB$. A number of other upper bounds can be computed. In the current version of GeoSteiner several upper bounds are computed — properly ordered so that the bounds that can be computed fast are tried first [391]. Even if $x$ cannot be pruned, it may be possible to reduce the feasible subarc. For example, assume that $UB \geq LB$, but $UB < |l(x)x|$ (and $UB \geq |r(x)x|$). We may now push $l(x)$ towards $r(x)$ on $\widehat{ab}$ until $|l(x)x| = UB$, in the same manner as for the bottleneck Steiner distance bound. The upper bound tests, however, can be quite time consuming, so it is important to find the right balance between running time and pruning efficiency.

### Construction and pruning of full Steiner trees

Full Steiner trees (FSTs) are constructed by combining two equilateral points/branches (lines 16-20 in Algorithm 1.3). More specifically, a valid FST $T$ is obtained if the Simpson line between the two equilateral points/branches $x_1$ and $x_2$ properly intersect their feasible subarcs. This does, however, result in multiple generations of $T$ — one for each edge of $T$. We may therefore arbitrarily select the edge of $T$ that is incident to the terminal spanned by $T$ that has the *highest index* in $N$. Thus, line 19 in Algorithm 1.3 first tests if $x_1$ is a terminal $t$ that has a higher index than all terminals in $N(x_2)$.

In summary, an FST $T$ is generated by combining a terminal $t = x_1$ with an equilateral point $x = x_2$. The Simpson line $tx$ must intersect the feasible subarc $\widehat{l(x)r(x)}$. The FST $T$ must be a minimum Steiner tree, and this can be tested by computing a number of heuristic trees interconnecting $\{t\} \cup N(x)$; if any of these heuristic trees has length less than $|T| = |tx|$, FST $T$ can be pruned. Again, it is important to find the right balance between running time and pruning efficiency.

### Numerical issues

The coordinates of the equilateral points can be written in the form $\alpha + \beta\sqrt{3}$, where $\alpha$ and $\beta$ are rational numbers [409, 391] (see also Section 1.2.1). The same holds for the endpoints of the feasible subarcs that are obtained in the projection test. More generally, trigonometric functions can be avoided in all the important pruning tests, and the coordinates involved are therefore algebraic numbers — and can in principle be represented symbolically with full precision.

In the GeoSteiner implementation an appropriate trade-off between precision and running time is chosen [391]. In the pure floating point version of the algorithm, all computations and comparisons are performed carefully to avoid numerical issues. It is also possible to compute equilateral points with full precision (in the form $\alpha + \beta\sqrt{3}$, where $\alpha$ and $\beta$ are rational numbers), but this has mainly been implemented to test the numerical robustness of the pure floating point version of the implementation.

### Use of data structures and overall performance of FST generation

When the GeoSteiner FST generation algorithm is used on large problem instances, say with more than 1000 terminals, the use of appropriate data structures becomes important.

Firstly, bottleneck Steiner distances between every pair of terminals can trivially be determined in $O(n^2)$ time by computing a minimum spanning tree and doing a depth-first traversal in this tree from every terminal. This gives constant-time lookup of bottleneck Steiner distances, but requires $\Theta(n^2)$ space. For large problem instances, it is more efficient to use the data structure described in [229] which uses $O(n)$ space; the preprocessing time is $O(n \log n)$, and queries can be made in $O(\log n)$ time.

The lune property test requires the identification of terminals within a certain region in the plane. So-called *range search* algorithms can be used for this purpose (see, e.g., [128]). In the GeoSteiner implementation a simple bucket structure has been applied. The minimum rectangle containing the set of terminals $N$ is divided into $K \times K$ subrectangles, and the terminals in each subrectangle are identified. This bucket structure enables fast range search in practice.

Althaus [12] has suggested the use of range search for identifying relevant pairs of equilateral points/branches. Let $B_{max}$ be the maximum bottleneck Steiner distance between any pair of terminals in $N$. Then no edge in any minimum Steiner tree for $N$ can be longer than $B_{max}$. Consider two equilateral points $a$ and $b$. Let $d_{ab}$ be the minimum distance between their feasible subarcs; if $a$ and $b$ are terminals, $d_{ab}$ is simply the distance between $a$ and $b$. Clearly, if $d_{ab} > 2B_{max}$ then no feasible combination of $a$ and $b$ exists, since at least one of the new edges would have a length greater than $B_{max}$. Thus, given equilateral point $a$, we can use the bucket data structure to identify all equilateral points of a given size — or branches spanning a given number of terminals — that are within distance $2B_{max}$ of $a$, where distance is defined as above. This method significantly improves the running time for large problem instances.

The practical performance of the GeoSteiner FST generation algorithm for the Euclidean Steiner tree problem has been carefully studied in [390, 409]. A number of improvements have been made since these papers were published around year 2000. Using the current version of the code on a modern computer, the FSTs for 1000 uniformly distributed terminals can be generated in less than 10 seconds. This amazing performance comes in particular from an efficient implementation of the lune property and bottleneck Steiner distance tests. For example, when combining equilateral points of size 1 (i.e., terminals), there exist $n(n-1)$ possible ordered pairs. For $n = 1000$, this yields almost one million possible pairs. The GeoSteiner algorithm combines these pairs in less than one second, and only around 5000 equilateral points of size 2 survive the pruning test (that is, on average each terminal participates in 10 equilateral points of size 2). For $n = 1000$, in total around 60000 equilateral points are generated, the largest having around 10–15 terminals, and around 2500 FSTs are generated. The running time is less than quadratic in practice: for $n = 10000$, the running time on the same modern computer is less than 500 seconds; the number of surviving equilateral points is around 600000 and the number of surviving FSTs is 26000.

### 1.4.4 Concatenation of full Steiner trees

The output of the FST generation phase is a set of FSTs $\mathcal{F} = \{T_1, T_2, \ldots, T_m\}$ that is guaranteed to contain the full components of at least one minimum Steiner tree for $N$. In the FST concatenation problem, we need to identify a subset $\mathcal{F}^* \subseteq \mathcal{F}$ such that $\mathcal{F}^*$ interconnects $N$ and has minimum total length. This problem can be solved in (at least) two ways: as a minimum spanning tree problem in hypergraphs or as a Steiner tree problem in graphs.

#### The FST concatenation problem as a minimum spanning tree problem in hypergraphs

The FST concatenation problem can be solved as a purely combinatorial problem if we think of each FST $T_i$ as a subset of terminals $N(T_i)$ having an associated length $|T_i|$; that is, we reduce the problem to an abstract combinatorial problem where the underlying geometry is completely ignored. The problem reduces to the so-called *minimum spanning tree problem in hypergraphs* (see Section 5.2.1 in Chapter 5). The vertex set of the hypergraph is the set of terminals $N$, and each FST is a weighted hyperedge. The task is to find a set of hyperedges that forms a spanning tree in the hypergraph, and that has minimum total weight. An integer programming algorithm [390] for the minimum spanning tree in hypergraphs problem can solve a typical 1000 terminal problem in a few seconds on a modern computer; however, the running time increases quickly and rather unpredictably for problems of size 10000 and greater.

#### The FST concatenation problem as a Steiner tree problem in graphs

The FST concatenation problem can also be formulated as an ordinary *Steiner tree problem in graphs* (see Section 5.1 in Chapter 5). The vertex set of the graph is the union of the terminals $N$ and the Steiner points in the set of generated FSTs $\mathcal{F}$. The edge set of the graph is the union of all edges in the generated FSTs, and the weight of each edge is the Euclidean length of the edge in the FST. The advantage of this approach is that well-engineered algorithms, also based on integer programming, already exist for the Steiner tree problem in graphs [314]. The disadvantage is that the approach breaks up the FSTs — making it feasible, in principle, to choose a subset of edges of a single FST. Another disadvantage is that the number of decision variables increases. Polzin and Vahdati Daneshmand [313] have studied the Steiner tree in graph approach from a theoretical and practical perspective, and have shown that the approach has significant potential; a benchmark of large-scale FST concatenation problems was in fact solved faster by this method than by using the minimum spanning tree in hypergraph integer programming approach.

## 1.5 Efficient constructions for special terminal sets

To conclude our study of Euclidean minimum Steiner trees, we look at families of sets consisting of special geometric configurations of terminals for which the problem can be solved in polynomial time. Such sets are of interest for a number of reasons: since the Euclidean Steiner tree problem is NP-hard, these sets give some insight into where the boundary between polynomial complexity and NP-hardness lies for this problem; also, such sets are useful for testing both exact and heuristic algorithms for the Steiner tree problem.

The proofs that minimum Steiner trees for particular families of terminal configurations can be constructed in polynomial time tend to be long and technical. Because of this, we will survey these results and the main underlying ideas, rather than giving all the details. We will also focus mainly on some of the most recent results from after the mid-1990s, since the cases from earlier in the literature have already been well surveyed in Part I, Chapter 5 of [213].

### 1.5.1 Terminals constrained to circles or curves

Let $N$ be the set of vertices of a regular $n$-gon. It seems intuitively reasonable that for sufficiently large values of $n$ any minimum Steiner tree for $N$ should coincide with a minimum spanning tree (MST) for $N$, in other words, should be composed of all except one of the edges of the regular $n$-gon. This problem was first studied in 1934 by Jarník and Kössler [225] and fully solved in 1987 by Du et al. [139] to give the following result.

**Theorem 1.19** *A minimum Steiner tree for the vertices of a regular $n$-gon, with $n \geq 6$, is an MST.*

Note that the vertices of a regular $n$-gon form a cocircular set. In 1992 Rubinstein and Thomas [330] used variational techniques to extend this theorem to more general sets of cocircular points (thus proving a conjecture of Graham [184]).

**Theorem 1.20** *Let $N$ be a set of cocircular teminals, lying on a circle of radius $r$. If at most one pair of adjacent terminals in $N$ have a distance between them greater than $r$, then a minimum Steiner tree for $N$ is an MST.*

In 1987, Ron Graham conjectured that the Steiner tree problem for any set of terminals lying on a circle could be solved in polynomial time. We now consider a more general version of this question: finding a minimum Steiner tree interconnecting a set of given points lying on a fixed set of smooth disjoint curves of finite total length in the plane. It was established by Rubinstein et al. [333], in 1997, that this problem can be solved in polynomial time, showing that Graham's conjecture also holds.

**Theorem 1.21** *[333] If $G$ is a fixed finite set of disjoint compact simple smooth curves in the plane, then there is a polynomial-time algorithm for the Euclidean Steiner tree problem where all terminals are constrained to lie on $G$.*

Note that a compact curve must have finite length; hence, the total length of $G$ is finite. The theorem ignores the complexity of the representation of $G$ and the terminals and assumes that elementary geometric constructions can be performed in bounded time. In particular, 'polynomial time' here means time polynomial in the number of terminals. Furthermore, the degree of the polynomial involved in the proof is in general quite high, and is highly dependent on the geometry of $G$. The running time of the algorithm depends on such features as the length and the maximum curvature of $G$ as well as the mutual proximity of curves or sections of curves in $G$.

We now give a brief outline of a strategy for proving Theorem 1.21, based on the proof in [333]. The underlying idea is to show that the number of topologies that need to be considered in order to find a minimum Steiner tree topology is a polynomial function of the number of terminals. The theorem then follows by applying the Melzak-Hwang algorithm (Theorem 1.5).

The main steps for restricting the number of possible topologies are as follows.

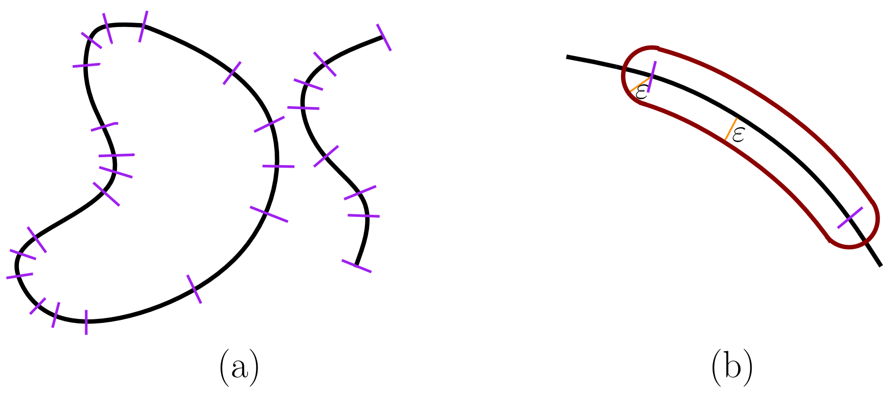


Figure 1.32: A covering $\mathcal{Q}$ for $G$, and a capsule $C(Q)$ for some $Q \in \mathcal{Q}$.

**Step 1: Construction of capsules**
Suppose $G$ is a fixed finite set of disjoint smooth (i.e., continuously differentiable) simple, compact curves of finite total length. In view of the smoothness of the curves of $G$, we can choose, for any $\delta > 0$, a covering of $G$ consisting of a finite set $\mathcal{Q} := \mathcal{Q}(\delta)$ of simply connected compact curves with the following properties:

1. elements of $\mathcal{Q}$ overlap only on their endpoints;
2. $G = \bigcup_{Q \in \mathcal{Q}} Q$; and
3. all tangents to the curve of $G$ at points in any fixed $Q \in \mathcal{Q}$ and in the (at most two) neighbouring curves in $\mathcal{Q}$ have orientations within $\delta$ of each other.

An example is shown in Figure 1.32(a).

By choosing a small enough $\delta$ we can consider the curves of $G$ to be approximately straight in each $Q \in \mathcal{Q}$ and its immediate neighbouring curves. We next define regions associated with the elements of $\mathcal{Q}$. If, for some $\varepsilon > 0$, the set of points at distance exactly $\varepsilon$ from the curves is self-intersecting, then there is a minimum strictly positive value at which this occurs. We will fix $\varepsilon$ to have a smaller positive value than this, as determined below.

Given $Q \in \mathcal{Q}$, we define the *capsule* $C(Q)$ to be

$$\{x : d(x, Q) \leq \varepsilon\},$$

as illustrated in Figure 1.32(b). Note that $G \cap C(Q)$ is a simply connected curve by the choice of $\varepsilon$.

We define the *orientation* of $G$ in a capsule to be the orientation of a tangent to $G$ at some point in the capsule. We choose $\varepsilon$ so small that the orientation of $G$ is within $\delta$ of the orientation of all tangents to the curve of $G$ at points within the capsule. This can be done by choosing it small enough to make sure that the parts of $G$ contained in the capsule are within $Q$ and the neighbouring elements of $\mathcal{Q}$. Note that, despite this condition on $\varepsilon$, we can still make $\delta$ as small as we please and (if necessary by subdividing elements in $\mathcal{Q}$) we can ensure that for each $Q \in \mathcal{Q}$ the length of $G \cap C(Q)$ is at most some bounded multiple of $\varepsilon$ (for example $100\varepsilon$).

The set of capsules may now be regarded as fixed for a given $G$. The determination of the capsules takes constant time independent of the number of terminals $n$, and hence does not contribute to the computational complexity of an algorithm for computing a minimum Steiner tree.

**Step 2: Bounding the number of full components that cross capsule boundaries**
Given a set $N$ of $n$ terminals lying on $G$, let $T^*$ be a minimum Steiner tree for $N$. Note that there is an upper bound $L$ on the length of $T^*$, independent of $N$. For example, we could take $L$ to be the total length of the curves in $G$ plus a sum of distances between the components of $G$.

Consider a full component $T$ of $T^*$ that interconnects some terminal on $Q \in \mathcal{Q}$ with some terminal not on $Q$. The intersection of $T$ and $C(Q)$ is a forest. A tree $T'$ in this forest is called a *terminating partial full* component (TPF component) if $T'$ connects to some terminal on $Q$; note that a TPF component by definition also touches one or more boundary points of $C(Q)$.

**Lemma 1.22** *The total number of TPF components in $T^*$ is bounded. Furthermore, the total number of points that TPF components touch on capsule boundaries is bounded.*

**Proof.** Let $T$ be a full component in $T^*$, and let $T'$ be one of the TPF components of $T$ in capsule $C(Q)$. The TPF component $T'$ connects some terminal on $Q$ with some point on the boundary of the capsule $C(Q)$; therefore, the length of $T'$ is at least $\varepsilon$. Since the length of $T^*$ is bounded by $L$, the total number of TPF components is bounded.

For the second part of the theorem, let $X$ be the set of points that $T'$ touches on the boundary of $C(Q)$; each point in $X$ corresponds to an intersection point between $T$ and the boundary of capsule $C(Q)$. The forest $T \setminus T'$ contains exactly $|X|$ trees. Each of these trees contains at least one TPF component — otherwise $T$ is not length-minimal. Therefore, the forest $T \setminus T'$ has length at least $|X|\varepsilon$. Again, since the length of $T^*$ is bounded by $L$, the size of set $X$ is bounded. Since the total number of TPF components is bounded, the total number of points that TPF components touch on capsule boundaries is also bounded. ∎

**Step 3: Enumerating all relevant TPF component topologies in polynomial time and constructing a minimum Steiner tree**

In this section we briefly argue that it is possible to enumerate all relevant TPF component topologies in polynomial time (and that the resulting number of TPF topologies is polynomial). Assume for a moment that this indeed is possible. Then a tree interconnecting all terminals can be constructed as follows:

1. Choose a bounded set of TPF topologies among the polynomial set of alternatives that is enumerated.
2. Choose a topology that interconnects the set of chosen TPF topologies.
3. Greedily add terminal-terminal connections between the remaining connected components.
4. For each full topology, locate the Steiner points using the Melzak-Hwang algorithm.

The total running time to try all possible choices — and hence to construct a minimum Steiner tree — is polynomial. Step 1 takes polynomial time since the number of bounded sets in a ground set of polynomial size is polynomial. Step 2 takes constant time since the total number of edges leaving the chosen set of TPF topologies is bounded; furthermore, the number of topologies on a bounded set is bounded. Step 3 takes polynomial time as it corresponds to running a minimum spanning tree algorithm on the set of terminals. Finally, Step 4 takes linear time per full topology. In conclusion, we can construct a minimum Steiner tree in polynomial time.

In order to complete the argument, we need to show that all relevant TPF topologies can be enumerated in polynomial time. Consider a capsule $C(Q)$. The challenge is that a single TPF topology in $C(Q)$ can span an unbounded number of terminals. On the other hand, $Q$ is close to straight. It is shown in [333] that a full component can only contain an unbounded number of Steiner points if it forms an *alternating branch* (as in Figure 1.33). An alternating branch is defined to be a full component in which one of the three orientations (say the red direction) is within $\delta$ of the orientation of $G$ in $C(Q)$, and the Steiner points all lie on a zigzag path using the other two orientations. An alternating branch contains a unique cherry; let $s_1, s_2, s_3, \ldots, s_j$ be the Steiner points in order along the zigzag path, starting at the cherry. In order for the alternating branch to be part of a minimum Steiner tree, no pair of red edges can run too close to each other, but at the same time each red edge must reach a terminal on $Q$. Since $Q$ is arbitrarily close to being straight, it follows that there is an arbitrarily large constant $\omega$ such that

$$\frac{d(s_i, s_{i+1})}{d(s_{i-1}, s_i)} > \omega$$

for $2 \leq i \leq j-1$. This inequality places strong geometric constraints on the alternating branch. In particular, it can be used to deduce two things: first, the terminals of two

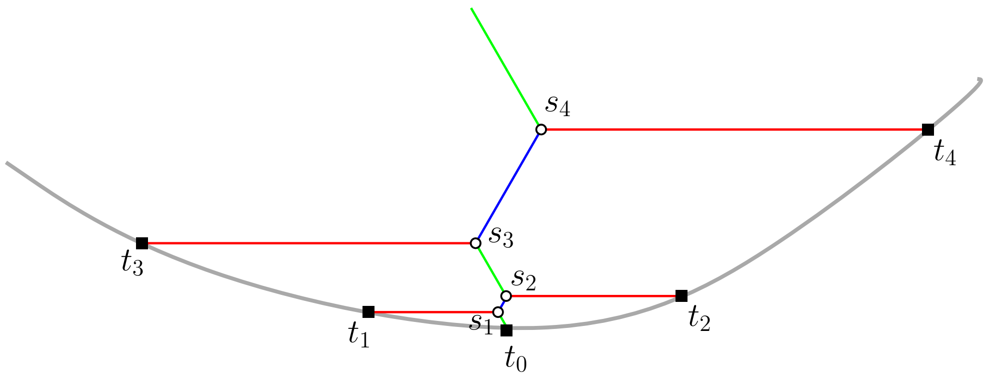


Figure 1.33: An alternating branch for terminals $t_0, t_1, t_2, t_3, t_4, \ldots$ on $G$. (Note that the alternating branch illustrated here cannot be part of a minimum Steiner tree since, for example, the edge $t_3s_3$ could be replaced by the edge $t_3t_1$, which is clearly shorter. In general, however, alternating branches can be part of a minimum Steiner tree if each ratio $d(s_i, s_{i+1})/d(s_{i-1}, s_i)$ is sufficiently large.)

different alternating branches (on opposite sides of $G$) cannot intermingle along $G$ very much; and second, the choice of which terminals are to be included in the branch for a specific capsule has a polynomial number of alternatives; the terminals on the branches are completely determined by the choice of cherry. The details of these results are rather technical, and can be found in [333]. These properties suffice to complete the proof of Theorem 1.21.

## 1.5.2 Terminals on rectangular lattices

A somewhat more useful class of terminal sets for which there exist polynomial-time solutions to the Steiner tree problem is the class of rectangular lattices:

**Definition [Rectangular lattices]**: A set of $mn$ terminals arranged in an $m \times n$ regular lattice of unit squares (like the corners of the cells of a checkerboard) is said to be an $m \times n$ *rectangular lattice* (or *square lattice* if $m = n$).

Unlike the polynomial-time algorithm for terminals constrained to curves in Section 1.5.1, where the polynomial-time algorithm has very high degree and is probably impractical to implement, the length of a minimum Steiner tree for a rectangular lattice can be computed in constant time, and the minimum Steiner tree itself is easy to construct.

The Steiner tree problem for rectangular lattices was first considered in 1989 by Chung, Gardner and Graham in [104]; these authors examined what they described as the checkerboard problem, that of solving the Steiner tree problem when the terminals form a square lattice. They gave a number of constructions for various cases which they

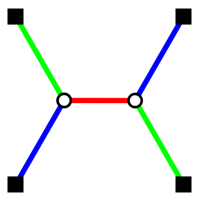


Figure 1.34: The minimum Steiner tree $X$.

conjectured to be minimal. These constructions use as a basic building block the Steiner tree for the corners of a unit square, which we will denote by $X$, shown in Figure 1.34.

The main constructions presented in [104] were (at the time) conjectured minimum Steiner trees for all $n \times n$ square lattices. These constructions fall naturally into six classes based on the value of $n \pmod 6$, except when $n$ is a power of $2$. The striking thing about all these constructions is that in each Steiner tree all but at most three of the full components are $X$'s and no full component contains more than $10$ terminals. This contrasts noticeably with the minimum Steiner trees for $2 \times n$ rectangular arrays, which have been shown in [105] to be full for all odd $n$.

In the case of a $2^k \times 2^k$ square lattice, there is a simple recursive construction for building a Steiner tree using only $X$'s. This is illustrated for the $8 \times 8$ square lattice in Figure 1.35. Note that the network is built from four networks for $4 \times 4$ square lattices connected in the centre by a single $X$. In a similar way a network composed only of $X$'s for a $2^k \times 2^k$ square lattice can be constructed from four networks for $2^{k-1} \times 2^{k-1}$ square lattices joined by a central $X$.

Chung et al. [104] showed that Steiner trees for rectangles in which all full components are $X$'s only occur for $2^k \times 2^k$ square lattices. Furthermore, they conjectured that these are also minimum Steiner trees. All of these conjectures (with a slight adjustment to one of the conjectured solutions) were proved in a series of papers by Brazil et al. [46, 58, 59]. The proof is long and technical, involving a lot of careful case analysis. Here we will simply outline the underlying idea and the nature of the results.

Let $T^*$ be a minimum Steiner tree for a rectangular lattice. A key to proving that the conjectured solutions are minimum is the observation that, per terminal, an $X$ appears to be the most efficient possible full component that can be part of $T^*$. This idea is formalised by the concept of *excess*, which we define below.

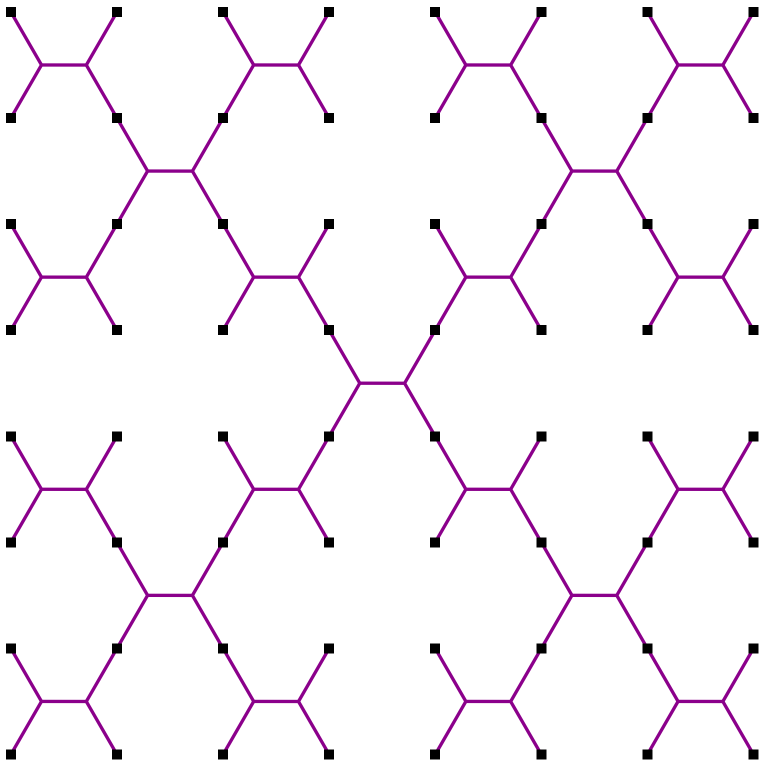


Figure 1.35: A Steiner tree for the $8 \times 8$ square lattice.

**Definition [Excess]**: Let $T'$ be a subtree of $T^*$ such that $T'$ is a union of full components of $T^*$ and spans $r$ terminals. Let

$$\rho = \frac{|X|}{3} = \frac{1+\sqrt{3}}{3} = 0.91068\cdots .$$

Then we define the *excess* of $T'$ to be

$$\mathbf{e}(T') = |T'| - (r-1)\rho.$$

Note that the excess is additive in the sense that if $T'$ is a subtree of $T^*$ such that $T' = \bigcup_{i=1}^{k} T_i$ where each $T_i$ is a full component of $T^*$, then $\mathbf{e}(T') = \sum_{i=1}^{k} \mathbf{e}(T_i)$. Also note that for any pair of Steiner trees on the same number of terminals the one with greater excess has greater length. By definition, $\mathbf{e}(X) = 0$.

The additive property of excess can be used to prove the following theorem.

**Theorem 1.23** *[46] Let $T$ denote a full component of a minimum Steiner tree $T^*$ for a rectangular lattice. Then either $T = X$ or $\mathbf{e}(T) > 0$.*

Let $S$ denote the set of connected components of the union of the vertices and edges of $T$ resulting when the boundaries of the cells of the rectangular lattice are deleted. Define the graph $G(T)$ to be the graph whose vertex set is $S$, two components in $S$ being adjacent in $G(T)$ if they both contain parts of the same edge of $T$ or if they both contain edges adjacent to a Steiner point on the boundary of a square. It is immediate that $G(T)$ is a tree since $T$ is a tree.

The proof of Theorem 1.23 works by induction on the number of terminals spanned by $T$ (say $m$) to show that certain graph structures for $T$ are impossible in or near the leaves of $G(T)$ if $\mathbf{e}(T) \leq 0$. It is assumed that $m$ is the smallest number of terminals for which there exists a full component $T$ such that $\mathbf{e}(T) \leq 0$, and then it is shown that this leads to a contradiction if $T \neq X$. To give a flavour of how this works, we prove a lemma below eliminating one of the possible graph structures of $T$ that could occur in a leaf of $G(T)$.

**Lemma 1.24** *Let $T$ denote a full component on $m$ terminals of a minimum Steiner tree $T^*$ for a rectangular lattice. Assume $\mathbf{e}(T) \leq 0$ and any connected graph on less than $m$ terminals has excess at least $0$. Then the part of $T$ occurring in a leaf of $G(T)$ cannot have the topology shown in Figure 1.36.*

**Proof.** Let $abcd$ be a cell of the rectangular lattice, and suppose the part of $T$ occurring in a leaf of $G(T)$ corresponding to this cell has the topology shown in Figure 1.36. Let $T_1$

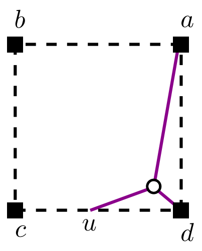


Figure 1.36: A topology for the part of $T$ corresponding to a leaf of $G(T)$.

denote the part of the branch of $T$ which lies above the line $cd$ and let $u$ be the point where $T$ intersects the interior of $cd$. Let $x$ be the distance from $d$ to $u$. Since $|T| \leq (m-1)\rho$, then by the minimality of $T$ we require that $|T_1| \leq x+\rho$, or in other words that $|T_1|-x \leq \rho$, for otherwise we can remove $T_1$ from $T$ and replace it by $du$. This would reduce the excess of the tree and the number of terminals, thereby contradicting the assumption in the lemma.

Let $Y$ denote the minimum Steiner tree on terminals $a, c, d$. A simple calculation (using the Melzak-Hwang algorithm) shows that $|T_1| = \sqrt{x^2 - x\sqrt{3}+1}$. The function $\sqrt{x^2 - x\sqrt{3}+1} - x$ is monotone decreasing for $x > 0$. But when $x = 1$,

$$|T_1| - x = |Y| - 1 = \frac{1+\sqrt{3}}{\sqrt{2}} - 1 = 0.93185\cdots > \rho,$$

which gives the required contradiction. ∎

Two other comparable results are given in Exercise 1.16. A series of similar but more sophisticated arguments allows one to eliminate all possible topologies for a leaf (or leaf and adjacent vertex) of $G(T)$, completing the proof of Theorem 1.23. The details of the proof are given in [46]. The following is an immediate corollary of this theorem:

**Corollary 1.25** *A minimum Steiner tree for a $2^k \times 2^k$ square lattice is composed entirely of $X$'s.*

Since $2^k \times 2^k$ square lattices are the only rectangular lattices for which there exist Steiner trees composed only of $X$'s, it follows that for any other rectangular lattice a minimum Steiner tree $T^*$ must have one or more full components not equal to $X$. In [58], a number of detailed geometric arguments are used to completely classify all possible full components of a minimum Steiner tree for a rectangular lattice, or indeed any nicely clustered set of lattice points. This is clearly an infinite set, since, for example, the minimum Steiner trees for $2 \times n$ rectangular lattices are full whenever $n$ is odd. The important property of this set is that the candidate full components can be divided into four distinct infinite families, each of which is easily described and constructed, and each of which

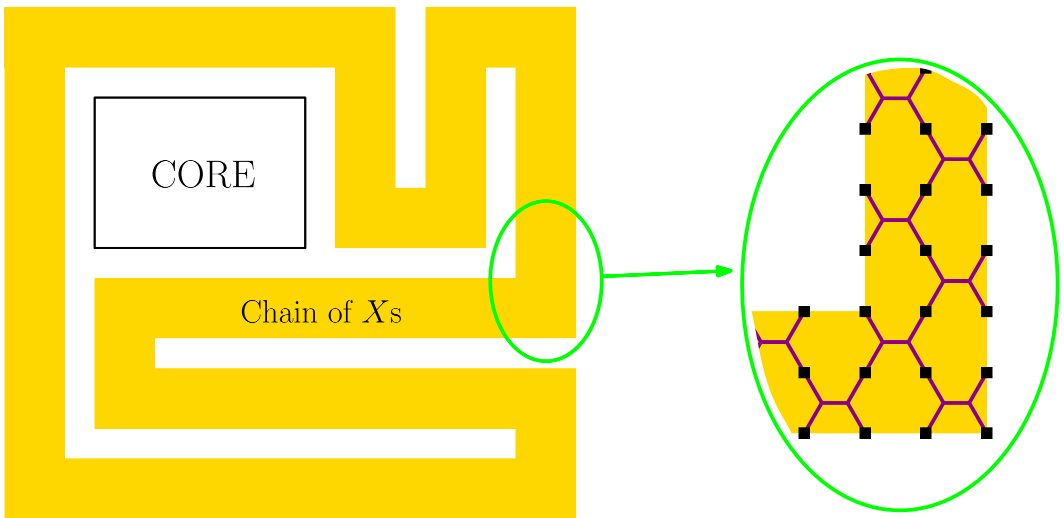


Figure 1.37: A central core and surrounding chain of $X$'s for an $m \times n$ rectangular lattice when $m$ and $n$ are both even. A detail of the chain of $X$'s is shown on the right.

has the property that the excess for the trees in the family monotonically increases (in an unbounded way) with the number of terminals. This means that only full components on a small number of terminals have small excess. Hence, we would expect a minimum Steiner tree for a rectangular lattice to be composed chiefly of $X$'s along with a small number of other small full components.

This complete classification of possible full components essentially turns the problem of constructing minimum Steiner trees for rectangular lattices into a purely combinatorial problem, which is solved in [59]. The main strategy is as follows. Let $T^*$ be a minimum Steiner tree for an $m \times n$ rectangular lattice. When $m$ and $n$ are large (i.e., greater than 7) we can think of $T$ as consisting of a chain of $X$'s winding around a central core, which is a minimum Steiner tree for an $a \times b$ rectangular array ($a$ and $b$ even) connected to the chain in a suitable way that makes the whole network a tree. This idea is illustrated in Figure 1.37, for $m$ and $n$ both even. If $m$ or $n$ is odd, we can delete the part of the chain of $X$'s running along the bottom or leftmost edge of the lattice. This construction gives a Steiner tree whose excess is small and is determined by the values of $m$ and $n \pmod 6$. Such a tree can be shown to be minimum by showing that the non-$X$ components cannot be replaced by any of the small number of other possible combinations of non-$X$ components with smaller excess.

In a similar way, minimum Steiner trees for $m \times n$ rectangular lattices where $m = 3, 5, 6$ or 7 have bounded excess and can be constructed by expanding a rectangular core using only $X$'s. The only remaining cases are the $2 \times n$ and $4 \times n$ rectangular lattices. Both of these cases have unbounded excess as $n$ increases, and hence require somewhat more sophisticated arguments to determine and validate the minimum networks. The $2 \times n$ case was studied by Chung and Graham [105], who showed that in this case the minimum Steiner tree is a single full component if $n$ is odd, or an alternating sequence of $X$'s and unit length edges if $n$ is even. For the $4 \times n$ case the minimum Steiner trees are composed of unit edges, about five times as many $X$'s and up to two other small full components. The strategy for proving minimality in this case is to show that the average per-column increase in excess for these trees is smaller than that for any other possibility. This involves some detailed case analysis, which again appears in [59].

## 1.6 Steiner trees in Minkowski planes

The previous sections of this chapter have presented properties of minimum Steiner trees in the Euclidean plane that derive from Euclidean geometry. The Euclidean plane is the most familiar example of a Minkowski or normed plane. In such a plane, the norm $\|\cdot\|$ is specified by $\mathcal{C}$, the boundary of a compact, convex, centrally symmetric domain, where $\mathcal{C} = \{x : \|x\| = 1\}$. In the Euclidean plane $\mathcal{C}$ is the Euclidean unit circle. See [276] and [370] for more details on Minkowski planes.

In terms of both theory and applications, there is much to be gained from understanding properties of minimum Steiner trees in Minkowski planes beyond the Euclidean plane. Chapters 2 and 3 both investigate properties of minimum Steiner trees in specific (non-Euclidean) families of norms, with applications to VLSI physical design. In this section, we look at what geometric properties of minimum Steiner trees can be established in general Minkowski planes, with as few restrictions on the unit circle $\mathcal{C}$ as possible. This will provide a generalisation of some of the Euclidean properties discussed earlier in this chapter, and will act as a toolbox of general properties which will prove useful in the next chapters. Most of the emphasis here will be on properties of Steiner points and Steiner configurations (which we define below).

We first establish some terminology and notation. In formal terms, the Steiner tree problem can be stated as follows:

MINKOWSKI STEINER TREE PROBLEM IN THE PLANE
**Given**: A set of points $N = \{t_1, \ldots, t_n\}$ lying on a Minkowski plane with unit circle $\mathcal{C}$.
**Find**: A geometric network $T = (V(T), E(T))$, such that $N \subseteq V(T)$, and such that $\sum_{e \in E(T)} \|e\|$ is minimised.

As in the Euclidean case, the initial set of points $N$ is referred to as a set of *terminals*; and any network solving the Minkowski Steiner tree problem is a tree, which we refer to as a *minimum Steiner tree*; elements of $V(T)$ not in $N$ are referred to as *Steiner points*. All non-zero edges in a minimum Steiner tree are geodesics between their endpoints and hence can be embedded as line segments in the given Minkowski plane. However (depending on the nature of the unit circle $\mathcal{C}$), there may be many other possible embeddings of a minimum edge, such as a zigzig path, since the Minkowski norm is not necessarily strictly convex.

For the Minkowski Steiner tree problem we define Steiner topologies and Steiner trees in exactly the same way as in Section 1.1.3. That is, a *Steiner topology* is a topology that can be realised by some non-degenerate minimum Steiner tree, and a *Steiner tree* is a non-degenerate relatively minimal tree for a Steiner topology.

Again, as in the Euclidean case, a Steiner tree can be decomposed into components in which every terminal is a leaf, known as *full components*, or *full Steiner trees*. This decomposition is unique for a given Steiner tree but is not necessarily unique for a given terminal set. Indeed, the number of full components may not be unique for a given terminal set. Because of this, we have the following definition:

**Definition [Fulsome trees]**: A Steiner tree $T$ is said to be *fulsome*[16] if it has the maximum possible number of full components amongst all Steiner trees with the same length as $T$ for the given terminal set. Hence, a Steiner tree is full and fulsome if there is no Steiner tree with the same length on the same set of terminals with two or more full components.

We note that even in the Euclidean plane full minimum Steiner trees are not necessarily fulsome (see Exercise 1.17). Restricting our attention to fulsome Steiner trees means that the structure of each full component is as simple as possible; it also helps in the development of good canonical forms.

**Definitions [Configurations]**: A *Steiner configuration* in a Minkowski plane is defined as a star with centre $s$ and leaves $x_1, \ldots, x_m$ (with $s, x_1, \ldots, x_m$ all distinct) that is part of some minimum Steiner tree with Steiner point $s$. If a star is not necessarily part of some minimum Steiner tree, then it is simply referred to as a *configuration*.[17]

Note that any configuration with centre $s$ that is part of a Steiner tree with Steiner point $s$ is, in fact, a Steiner configuration (Exercise 1.18).

The results in this section are mainly concerned with local properties of a Steiner tree in a general Minkowski plane; more specifically, with Steiner configurations. As discussed in the previous sections, the Steiner point can be viewed as a solution to the general Fermat-Torricelli problem for its neighbouring points $x_1, \ldots, x_m$ — i.e., as a Fermat-Torricelli point. A useful survey on results pertaining to this problem in general Minkowski planes and spaces is given by Martini et al. [277].

Many of the geometric properties developed in this section are largely a consequence of the fundamental properties of a norm, such as the triangle inequality and positive definiteness. This is illustrated in the proof of the following useful theorem.

**Theorem 1.26** ***[Pointed Configuration Theorem]*** *Let $\{sx_i : i = 1, \ldots, m\}$ be a configuration about $s$ in a Minkowski plane. If there is a line $L$ through $s$ such that the interior of each segment $sx_i$ is in the same open half-plane bounded by $L$, then $\{sx_i\}$ is not a Steiner configuration.*

**Proof.** We first consider the case where $m = 3$. We may assume, without loss of generality, that the points $x_i$ are labelled so that $x_2$ lies in the convex cone bounded by $\overrightarrow{sx_1}$ and $\overrightarrow{sx_2}$. Hence, there exist points $x'_1$ in the interior of $sx_1$ and $x'_3$ in the interior of $sx_3$ such that the line segment $x'_1x'_3$ intersects the interior of $sx_2$, at say $x'_2$. By the triangle inequality, $\|sx'_1\| + \|sx'_3\| \geq \|x'_1x'_3\|$, and by positive definiteness, $\|sx'_2\| > 0$. Hence, replacing the line segments $\{sx'_i : i = 1, \ldots, 3\}$ by $x'_1x'_3$ strictly reduces the length of the configuration (while maintaining connectivity), showing that $\{sx_i\}$ is not a Steiner configuration.

A similar argument applies if $m > 3$ (Exercise 1.19). ∎

### 1.6.1 Steiner points of degree $3$

We have seen in Section 1.1 above that in the Euclidean plane every Steiner point of a Steiner tree has degree $3$. This result generalises to any smooth Minkowski plane (a Minkowski plane is said to be *smooth* if every point on the corresponding unit circle $\mathcal{C}$ is differentiable).

**Theorem 1.27** *[243] If $T$ is a minimum Steiner tree in a smooth Minkowski plane, then every Steiner point of $T$ has degree $3$.*

For the proof of this theorem we refer the reader to [243] — the proof relies on a somewhat technical analytic argument.[18]

Theorem 1.27 does not hold in general Minkowski planes, where the boundary of the unit ball may have non-differentiable points. However, as we will see later, in these cases higher degree Steiner points only occur in very special circumstances. In this subsection we focus on properties of degree $3$ Steiner configurations in Minkowski planes. The most important of these results is the centroid theorem (Theorem 1.28 below), which will also be used extensively in Chapter 2.

Du et al. [134] have proved the following result for strictly convex and differentiable unit circles (which includes the Euclidean metric): if one of the edges of a degree $3$ Steiner

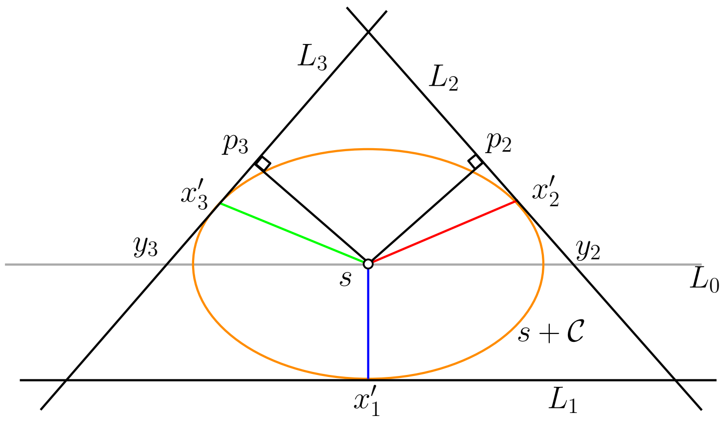


Figure 1.38: The geometry around a Steiner point $s$.

configuration, say $sx_1$, is given, then the orientations of the other edges $sx_2$ and $sx_3$ are *unique*. In other words, if the Steiner point $s$ and one of the leaves, say $x_1$, are given, then the meeting angles of $s$ are unique. Below, we present a proof of this result and establish an analogous theorem for general unit circles.

Our approach here is based on properties established by Chakerian and Ghandehari [80] for norms with strictly convex and differentiable unit circles.

**Theorem 1.28** ***[Centroid Theorem]*** *Let $x_1$, $x_2$ and $x_3$ be a set of leaves of a Steiner configuration in the Minkowski plane (with unit circle $\mathcal{C}$) with Steiner point $s$. Let $x'_1$, $x'_2$ and $x'_3$ be the points where $s + \mathcal{C}$ intersects $\overrightarrow{sx_1}$, $\overrightarrow{sx_2}$ and $\overrightarrow{sx_3}$, respectively. Then for each $i \in \{1, 2, 3\}$ there exists a line $L_i$ which is a supporting line of $z + \mathcal{C}$ at $x'_i$, such that $L_1$, $L_2$ and $L_3$ form a triangle whose centroid coincides with $s$.*

**Proof.** [19] By the properties of the gradient of the gauge function in the Minkowski plane, it follows that there exist supporting lines $L_i$ of $s + \mathcal{C}$ at each $x_i$, such that

$$\frac{\mathbf{u}_1}{h_1} + \frac{\mathbf{u}_2}{h_2} + \frac{\mathbf{u}_3}{h_3} = 0 \tag{1.3}$$

where each $\mathbf{u}_i$ is the outward normal vector to the supporting line $L_i$, and each $h_i$ is the (Euclidean) distance from $s$ to $L_i$. This is illustrated in Figure 1.38. Since a strictly positive linear combination of the vectors $\mathbf{u}_i$ equals $\mathbf{0}$, it follows that $L_1$, $L_2$ and $L_3$ form a triangle around $s + \mathcal{C}$, which we denote by $\Delta$.

Let $L_0$ be the line through $s$ parallel to $L_1$. For $i \in \{2, 3\}$, let $p_i$ be the intersection of $L_i$ and the line perpendicular to $L_i$ through $s$, and let $y_i = L_i \cap L_0$. Note that $h_i = |sp_i|$. Equation (1.3) implies that $\mathbf{u}_2/h_2 + \mathbf{u}_3/h_3$ is perpendicular to $L_0$, and hence that

$$\frac{cos(\angle p_2 s y_2)}{|sp_2|} = \frac{cos(\angle p_3 s y_3)}{|sp_3|} \tag{1.4}$$

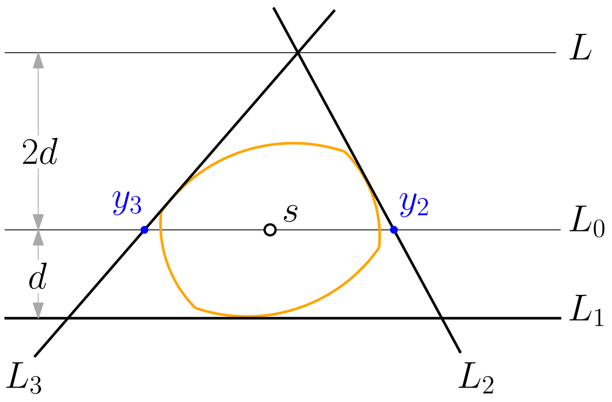


Figure 1.39: The centroid property. Here $s$ is the centroid of the triangle formed by $L_1$, $L_2$ and $L_3$ if $|sy_2| = |sy_3|$.

which implies $|y_2s| = |y_3s|$. In other words, $s$ lies on the median of $\Delta$ through $L_2 \cap L_3$ (recalling that a median of a triangle is a line segment joining a vertex of the triangle to the midpoint of the opposing side). By symmetric arguments, $s$ also lies on the other two medians of $\Delta$, and hence coincides with the centroid of $\Delta$. ∎

**Definition [Centroid property]**: Given a Minkowski unit circle $\mathcal{C}$, we say that any set of supporting lines of $\mathcal{C}$ forming a triangle whose centroid is the centre of $\mathcal{C}$ satisfies the *centroid property*.

In terms of the above definition, Theorem 1.28 shows that for any degree $3$ Steiner configuration at a point $s$ there exists a set of lines supporting $s + \mathcal{C}$ at points determined by the Steiner configuration that satisfies the *centroid property*.

A useful consequence of the centroid property is that in a smooth Minkowski plane the edges in a full Steiner tree use at most three directions. Before proving this theorem we first establish a series of lemmas characterising properties of sets of supporting lines satisfying the centroid property.

The first lemma follows immediately from the observation that the centroid of a triangle divides each of its medians in the ratio $2:1$.

**Lemma 1.29** *[134] Let $L_1$, $L_2$ and $L_3$ be three supporting lines of the unit circle $s + \mathcal{C}$, and let $L_0$ be the line that is parallel to $L_1$ and contains $s$. Let $d$ be the Euclidean distance between $L_1$ and $L_0$, and define $L$ to be the line that is parallel to $L_1$ at distance $3d$ from $L_1$, and at distance $2d$ from $L_0$ (as in Figure 1.39). Let $y_2 = L_2 \cap L_0$ and let $y_3 = L_3 \cap L_0$. Then supporting lines $L_1$, $L_2$ and $L_3$ satisfy the centroid property if and only if (i) $L_2 \cap L_3$ lies on $L$, and (ii) $|sy_2| = |sy_3|$.*

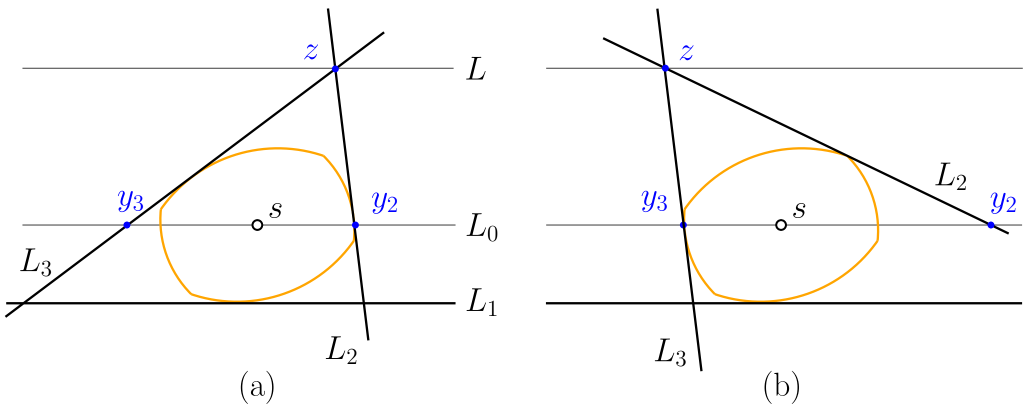


Figure 1.40: Illustration of proof of Lemma 1.31. (a) The initial positions of supporting lines $L_2$ and $L_3$. (b) The final positions of the supporting lines. The position for which the centroid property holds lies between these two extremes.

In the following lemmas we continue to refer to $L_0$ and $L$ as defined by Lemma 1.29. These two lemmas give some useful characterisations of sets of supporting lines that fulfil the centroid property.

**Lemma 1.30** *Let $L_1$, $L_2$ and $L_3$ be a set of supporting lines of $s+\mathcal{C}$ that fulfils the centroid property, and let $L_0$ be the line that is parallel to $L_1$ and contains $s$. Then neither $L_2$ nor $L_3$ supports $\mathcal{C}$ at a point that is strictly between $L_1$ and $L_0$.*

The proof of the above lemma is left as an exercise for the reader (Exercise 1.20).

**Lemma 1.31** *Let $L_1$ be a line that supports a unit circle $\mathcal{C}$ with centre $s$, and let $L_0$ be the line that is parallel to $L_1$ and contains $s$. If $L_0$ intersects $\mathcal{C}$ at a differentiable point, then there exists exactly one pair of supporting lines $L_2$ and $L_3$, such that $L_1$, $L_2$ and $L_3$ have the centroid property.*

**Proof.** Assume $L_0$ intersects $\mathcal{C}$ at a differentiable point. By the central symmetry of $\mathcal{C}$, both the points where $\mathcal{C}$ intersects $L_0$ are differentiable. Consider a line $L_2$ supporting $\mathcal{C}$ at one of these two intersection points (Figure 1.40(a)). Note that $L_2$ is in fact a tangent of $\mathcal{C}$.

Let $L$ be defined as in Lemma 1.30, and let $z$ be the intersection between $L_2$ and $L$. Define $L_3$ as the unique supporting line that contains $z$ and supports $\mathcal{C}$ on the ‘opposite’ side to $L_2$, such that $L_1$, $L_2$ and $L_3$ form a triangle with $s$ in its interior. Let $y_2$ (respectively $y_3$) be the intersection of $L_2$ (respectively $L_3$) with $L_0$ (Figure 1.40(a)). By Lemma 1.30 we only need to consider supporting lines that meet $\mathcal{C}$ on or above $L_0$.

Imagine rotating $L_2$ and $L_3$ jointly in a counter-clockwise manner around $\mathcal{C}$, such that they continuously intersect $L$ at a common point $z$. This rotation strictly increases $|sy_2|$, while it strictly decreases $|sy_3|$, since all rotation points are strictly above $L_0$. The rotation is performed until $L_3$ is parallel to the original line $L_2$ (and thus supports the opposite point of $\mathcal{C}$ that is also on $L_0$), as in Figure 1.40(b). Before the rotation started we clearly had $|sy_2| < |sy_3|$, and when the rotation ends we have $|sy_2| > |sy_3|$. Hence, since both distances are strictly increasing/decreasing, there exists exactly one point $z$ on $L$ where the corresponding supporting lines $L_2$ and $L_3$ have $|sy_2| = |sy_3|$. For this and only this set of supporting lines the centroid property holds. ∎

An immediate consequence of Lemma 1.31 in a smooth Minkowski plane is the following result:

**Corollary 1.32** *In a smooth Minkowski plane the directions of the edges in a degree* $3$ *Steiner configuration are uniquely determined by the direction of any one edge.*

**Theorem 1.33** *In a smooth Minkowski plane the edges of a full Steiner tree use at most three distinct directions.*

**Proof.** Let $T$ be a full Steiner tree in a smooth Minkowski plane. By Theorem 1.27 every Steiner point in $T$ has degree $3$. Since any pair of neighbouring Steiner points in $T$ share a common incident edge, it follows from Corollary 1.32 that the incident edges of each of those Steiner points each use the same set of directions. The theorem follows from the observation that the Steiner points of a full Steiner tree induce a tree. ∎

### 1.6.2 Steiner points of degree $\geq 4$

In this subsection we consider higher degree Steiner points in Minkowski planes which are not smooth. All of the results in this section are based on a *replacement principle*. This principle operates in one of two ways: either we replace certain line segments in a minimum tree $T$ by new line segments with the same length and direction (and hence the same cost), or we replace a set of line segments by a new set of line segments which by minimality we can show have the same cost. The result is to either find an alternative form for $T$, or in some cases to obtain a contradiction to the minimality or assumed fulsomeness of $T$.

We begin by showing that it suffices to investigate properties of degree $4$ Steiner points in a minimum Steiner tree.

**Theorem 1.34** *[107, 7] Given a set of terminals* $N$ *in a Minkowski plane, there exists a minimum Steiner tree* $T$ *for* $N$ *in which every vertex has degree at most* $4$.

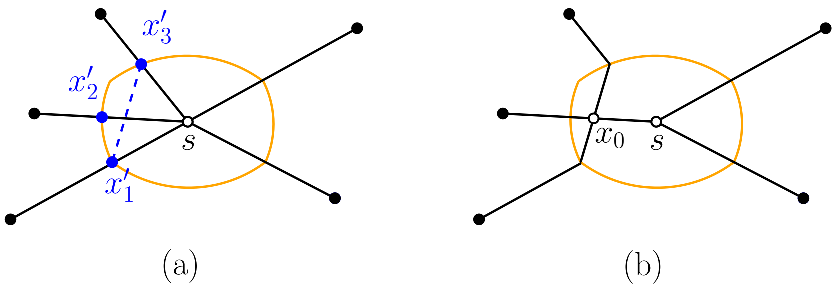


Figure 1.41: (a) An example of a Steiner point of degree 5; the Minkowski unit circle around $s$ is shown in orange. (b) The Steiner point $s$ can be split into two Steiner points as shown without increasing the length of the tree.

**Proof.** Let $s$ be a Steiner point of degree $5$ or more with five adjacent vertices $x_1, \ldots, x_5$. By rescaling if necessary, we can assume that for each $i \in \{1, \ldots, 5\}$ we have $\|sx_i\| > 1$. Let $x'_i$ be the point on the interior of the line segment $sx_i$ such that $\|sx'_i\| = 1$ (i.e., the points $x'_i$ lie on the Minkowski unit circle around $s$, as in Figure 1.41(a)). There exists a set of three points in $\{x'_i | i \in \{1, \ldots, 5\}\}$, say $x'_1, x'_2, x'_3$ in counter-clockwise order around the unit circle, such that all three points lie in the interior of a half-plane induced by a line through $s$. (See Figure 1.41(a).) By the convexity of the unit ball, it follows that $x'_1x'_3 \cap sx'_2 \neq \emptyset$. Let $x'_1x'_3 \cap sx'_2 := x_0$, and note that $x_0 \neq s$. By the triangle inequality $\|x'_1x'_3\| \leq \|sx'_1\| + \|sx'_3\|$. Hence, if we replace $\{sx'_1, sx'_3\}$ in $T$ by $\{x_0x'_1, x_0x'_3\}$ to form a new interconnection tree $T'$ (as illustrated in Figure 1.41(b)), then $\|T'\| \leq \|T\|$, and we have reduced the degree of $s$ (by 2) by introducing a new degree $4$ Steiner point $x_0$. This procedure can be repeated until every Steiner point has degree at most $4$. ∎

We note that Swanepoel [361] has strengthened this result, using properties of the dual and methods from functional analysis, to show that in a minimum Steiner tree, Steiner points of degree $5$ or more never occur (even without a fulsomeness condition).

In light of Theorem 1.34, we can now restrict our attention to Steiner points of degree $4$.

**Definition [Opposite pairs of edges]**: Let $\{sx_i : i = 1, \ldots, 4\}$ be a degree $4$ Steiner configuration around $s$ where the neighbours of $s$ are indexed in counter-clockwise order around $s$. We say that such a Steiner configuration consists of two *opposite pairs of edges*, $\{sx_1, sx_3\}$ and $\{sx_2, sx_4\}$.

**Lemma 1.35** *[361] In a degree $4$ Steiner configuration in a Minkowski plane one of the opposite pairs of edges is collinear.*

**Proof.** Let $\{sx_i : i = 1, \ldots, 4\}$ be a degree $4$ Steiner configuration around a Steiner point $s$ where the $x_i$ are indexed in counter-clockwise order around $s$. We assume that neither

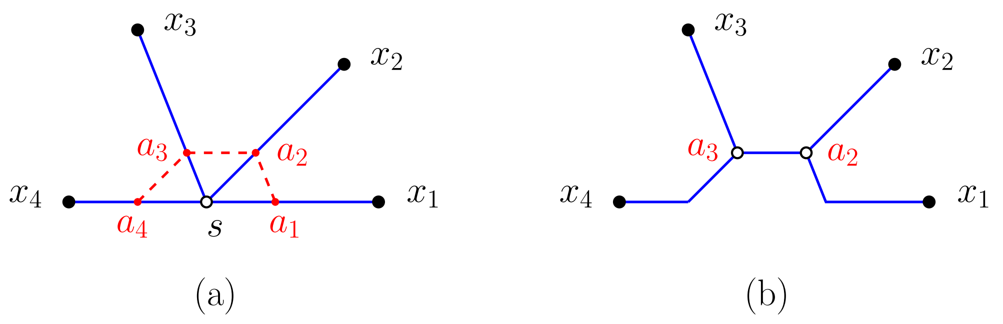


Figure 1.42: A configuration of degree $4$ with a meeting angle of $\pi$ is not a Steiner configuration. The Steiner point $s$ in (a) can be replaced by two Steiner points $a_2$ and $a_3$ in (b) to create a strictly shorter interconnection tree.

of the opposite pairs of edges is collinear, and obtain a contradiction. By Theorem 1.26 there is no line $L$ through $s$ such that the interior of each segment $sx_i$ is in the same open half-plane bounded by $L$. We now show that the same is true for a closed half-plane. Suppose, on the contrary, that $sx_4$ and $sx_1$ are collinear, as in Figure 1.42(a). Choose $a_i \in sx_i$ (for $i = 1, \dots, 4$) such that $a_1a_2 \parallel sx_3$, $a_2a_3 \parallel sx_1$ and $a_3a_4 \parallel sx_2$. If we replace the line segments $\{sa_i : i = 1, \dots, 4\}$ in the Steiner configuration by $\{a_1a_2, a_2a_3, a_3a_4\}$, then the resulting tree

$$\{x_1a_1, a_1a_2, x_2a_2, a_2a_3, x_3a_3, a_3a_4, x_4a_4\}$$

(illustrated in Figure 1.42(b)) still interconnects the terminals $x_i$, but is strictly shorter (by $\|sa_4\|$) than the Steiner configuration, contradicting minimality.

By relabelling the terminals if necessary, we may therefore assume, without loss of generality, that $x_3$ and $x_4$ lie on opposite sides of the line $\overline{sx_1}$ and $x_1$ and $x_2$ lie on opposite sides of the line $\overline{sx_4}$, as in Figure 1.43(a). Choose $b_i \in sx_i$ (for $i = 1, \dots, 4$) and $c \in sx_2$ such that $sb_1b_2b_3$ and $scb_3b_4$ are parallelograms. This is illustrated in Figure 1.43(a), where the new edges of the two parallelograms are coloured red and green, respectively. Note that since $x_1$ and $x_2$ lie on opposite sides of $\overline{sx_4}$ it follows that $b_2 \in sc$. Furthermore, since the configuration is minimum, we have that $\|b_3c\| \geq \|sc\|$, since otherwise we could reduce the length of the tree by introducing a second Steiner point at $b_3$. Hence $\|sb_4\| \geq \|b_3b_4\|$. It follows that the tree

$$\{x_1b_1, b_1b_2, x_2b_2, b_2b_3, x_3b_3, b_3b_4, x_4b_4\},$$

(shown in Figure 1.43(b)) interconnecting the terminals $x_i$, is shorter than the Steiner configuration (by at least $\|sb_2\|$), again contradicting minimality. ∎

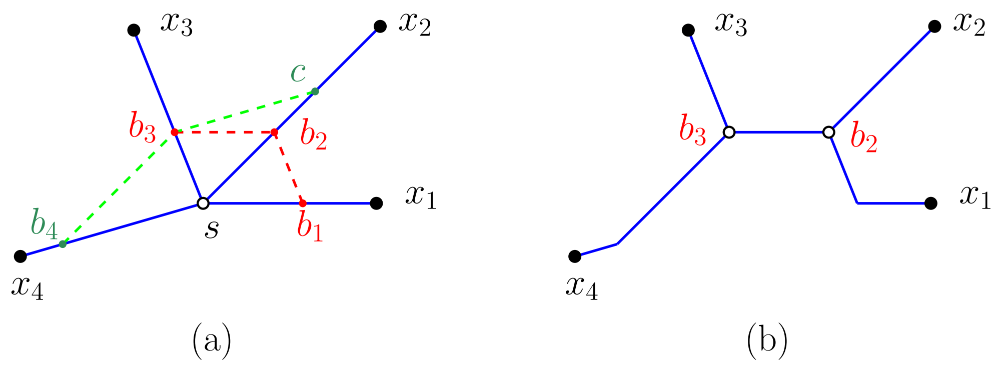


Figure 1.43: A configuration with no opposite pair of edges being collinear is not a Steiner configuration. The Steiner point $s$ in (a) can be replaced by two Steiner points $b_2$ and $b_3$ in (b) to create a strictly shorter interconnection tree.

**Definitions [First and second opposite pairs]**: Let $\{sx_i : i = 1, \ldots, 4\}$ be a degree $4$ Steiner configuration around $s$ where the neighbours of $s$ are indexed in counter-clockwise order around $s$. One of the opposite pairs of edges, say $\{(s, x_1), (s, x_3)\}$, must be collinear, by Lemma 1.35; we refer to this as the *first* opposite pair of edges. The other pair of edges, $\{(s, x_2), (s, x_4)\}$, is called the *second* opposite pair of edges.

The above classification into pairs is not necessarily unique, but this is not important in the following.

**Lemma 1.36** *Suppose the second opposite pair of edges $\{(s, x_2), (s, x_4)\}$ around a degree $4$ Steiner point $s$ in a Steiner configuation $\{sx_i : i = 1, \ldots, 4\}$ is not collinear. Then there exists a point $s'$ in the interior of $sx_1$ or $sx_3$ such that for every point $s_0 \in ss'$ we have $\|sx_2\| = \|s_0x_2\|$ and $\|sx_4\| = \|s_0x_4\|$.*

The proof of this lemma is straightforward and is left as an exercise for the reader (Exercise 1.21).

A consequence of the above lemma is that if the second opposite pair of edges is not collinear, then we can split the Steiner point $s$ into a pair of adjacent degree $3$ Steiner points, as illustrated in Figure 1.44. Let $T_a$, $T_b$ and $T_c$ be the configurations (all on the same set of terminals) shown in diagrams (a), (b) and (c) of the figure, respectively. Observe that $2\|T_a\| = \|T_b\| + \|T_c\|$; hence, if $T_a$ is a Steiner configuration, then $T_b$ and $T_c$ are each Steiner configurations as well. Note that this split can be made in two topologically different ways, as indicated by the different colourings of edges in Figure 1.44(b) and (c).

**Definition [Cross]**: A *cross* is defined to be a degree $4$ Steiner point where both the first and second opposite pairs of edges are collinear.

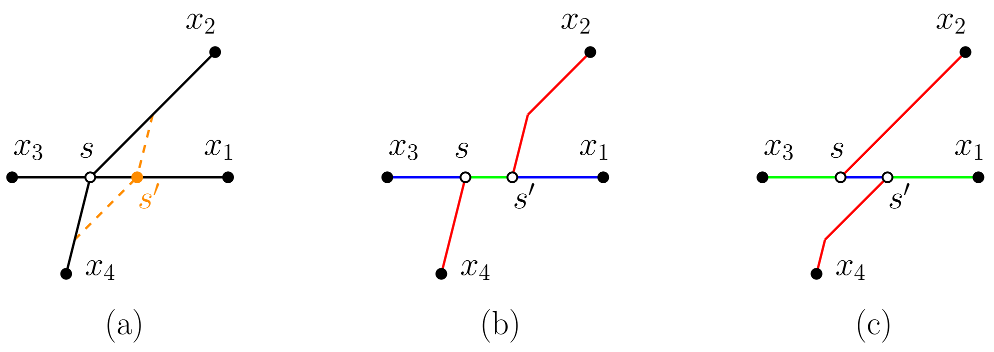


Figure 1.44: Diagram (a) shows a degree $4$ Steiner configuration where the second opposite pair of edges is not collinear. The Steiner point can be split into two degree $3$ Steiner points in two different ways, as shown in (b) and (c).

Thus far we have shown that unless a degree $4$ Steiner point is a cross, we can always split it into two adjacent degree $3$ Steiner points.

We now look at properties of a degree $4$ Steiner point in a full and fulsome component of a minimum Steiner tree. One of the main aims is to show that even if the Steiner point is a cross, then in most cases it can still be split into two degree $3$ Steiner points without increasing the length of the Steiner tree. A key lemma for helping establish this is the following.

**Lemma 1.37** ***[Sliding lemma]*** *Let $e = (s_1, s_2)$ be an edge connecting two Steiner points ($s_1$ and $s_2$) in a fulsome minimum Steiner tree $T$. Let $e_1 = (s_1, v_1)$ be the next edge incident with $s_1$ (in counter-clockwise order around $s_1$ from $e$), and let $e_2 = (s_2, v_2)$ be the next edge incident with $s_2$ (in clockwise order around $s_2$ from $e$) — see Figure 1.45. Then $\theta$, the angle at $s_1$ between $e$ and $e_1$, and $\phi$, the angle at $s_2$ between $e$ and $e_1$, satisfy $\theta + \phi > \pi$.*

**Proof.** Let $\mathbf{u}_1$ be the outward direction of $e_1$ at $s_1$ and let $\mathbf{u}_2$ be the outward direction of $x_2$ at $s_2$. Observe that

$$\theta + \phi \geq \pi \tag{1.5}$$

since otherwise we could simultaneously perturb $s_1$ in direction $\mathbf{u}_1$ and $s_2$ in direction $\mathbf{u}_2$, decreasing $\|s_1 s_2\|$ and hence decreasing $\|T\|$, which contradicts minimality.

Now suppose that the lemma does not hold; in other words, that $\theta + \phi = \pi$, as illustrated in Figure 1.45. Then we can perform a *slide* on $e$, by which we mean a simultaneous movement of $s_1$ in direction $\mathbf{u}_1$ and $s_2$ in direction $\mathbf{u}_2$ (at the same speed), without increasing the length of $e$.[20] We can continue to slide $e$ until either $s_1$ coincides with $v_1$ or $s_2$ coincides with $v_2$.

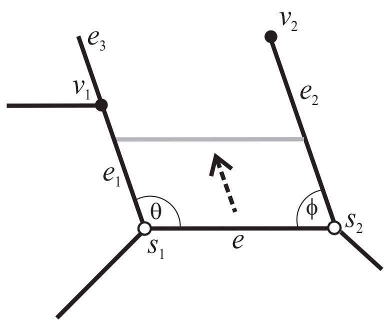


Figure 1.45: A slide on the edge $e$.

Assume, without loss of generality, that $s_1$ coinciding with $v_1$ occurs first. If $v_1$ is a terminal, then we have a contradiction to $T$ being fulsome. If $v_1$ is a Steiner point, then let $\theta_1$ be the meeting angle at $v_1$ between $e_1$ and the next edge $e_3$ going counter-clockwise around $v_1$ from $e_1$. By Theorem 1.26, $\theta_1 \le \pi$. If $\theta_1 < \pi$ then continuing to slide $e$ past $v_1$ strictly decreases the length of $e$, contradicting the minimality of $T$. Hence $\theta_1 = \pi$. We can now continue to slide $e$ past $v_1$ without increasing $\|T\|$. The same argument applies at each Steiner point encountered (or if two Steiner points are encountered simultaneously). Hence, we can continue the slide until we reach a terminal, giving a contradiction to $T$ being fulsome. ∎

Before applying the sliding lemma to degree $4$ Steiner points, we note the following result, which is a straightforward corollary.

**Lemma 1.38** *Let $s$ be a degree $3$ Steiner point in a fulsome minimum Steiner tree in a Minkowski plane. Let $u$, $v$ and $w$ be the three neighbouring nodes of $s$ in $T$, and suppose the edges $su$ and $sv$ are collinear. Then $w$ is a terminal.*

The proof of this lemma requires a straightforward argument by contradiction (Exercise 1.22).

**Theorem 1.39** *In a fulsome minimum Steiner tree, a degree $4$ Steiner point can always be split into two adjacent degree $3$ Steiner points* unless *it is a cross and is adjacent to terminals only.*

**Proof.** By the comment following Lemma 1.36, we only need to consider the case where the Steiner point $s$ is a cross. Assume that one of the neighbours of $s$ is a Steiner point $v$, and that $v$ has degree $3$. Let $a$, $b$ and $s$ be the neighbours of $v$ (as in Figure 1.46(a)).

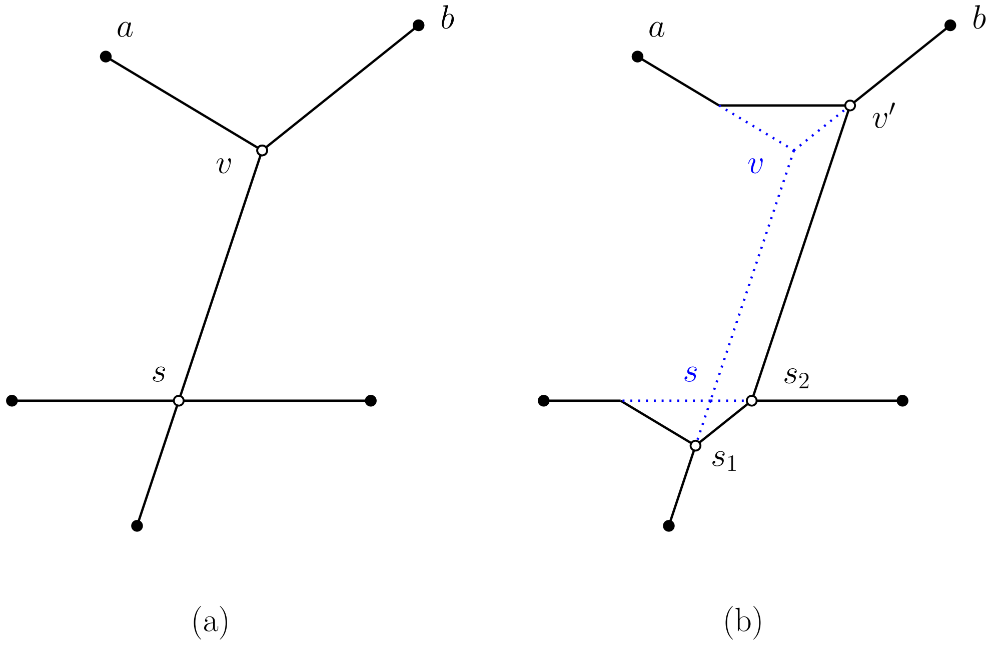


Figure 1.46: Splitting of a cross into two adjacent degree $3$ Steiner points. Diagram (a) shows a cross $s$ with an adjacent Steiner point $v$. Diagram (b) shows a local perturbation that splits $s$ into two degree $3$ Steiner points without increasing the length of the tree.

By Lemma 1.37, neither of the edges $va$ or $vb$ is parallel to the edges in the opposite pair at $s$ that does not include $vs$. Hence, we can make the (small) local change indicated in Figure 1.46(b) without increasing the length of the tree. This change splits $s$ into two adjacent degree $3$ Steiner points $s_1$ and $s_2$ while moving $v$ to a new position $v'$.

Finally, if $v$ is a degree $4$ Steiner point, then we arrive at a contradiction to length-minimality, since the local change given above would decrease the length of the tree. ∎

It follows that in a GeoSteiner-type algorithm for constructing a minimum Steiner tree one can, for the most part, limit the construction of candidate full Steiner components to full and fulsome minimum Steiner trees where all Steiner points have degree $3$. In the chapters that follow we will see that for many Minkowski metrics there exist efficient methods for constructing such candidates. (The construction of a cross with terminals as neighbours can easily be handled separately.)

Finally, we note that although the results in this section have been expressed in terms of properties of minimum Steiner trees, it is straightforward to see that they also apply to a general Steiner tree $T$: whenever a length-reducing replacement changes the topology in the above proofs it invalidates the underlying Steiner topology of $T$; when a length-reducing replacement does not change the topology it shows that $T$ is not relatively minimal. This observation will be useful in Chapter 2, where we apply a number of these results.

## 1.7 Applications and extensions

In this section we first discuss applications of the Euclidean Steiner tree problem both to practical engineering problems and to other areas of mathematics. We then look briefly at the extension of the problem to higher dimensional spaces.

### 1.7.1 Applications

The Euclidean Steiner tree problem in the plane is a problem that immediately gives the impression that it should have useful physical applications. Indeed, most of the earliest literature in this area poses the problem in terms of applications. The first known paper on the Euclidean Steiner tree problem, by the French mathematician Joseph Diaz Gergonne [175], addresses a formulation of the problem that involves designing a network of canals interconnecting a given set of cities (though in the solution the problem is treated as an abstract problem in pure geometry). When the problem was independently rediscovered some 25 years later by Carl Friedrich Gauss in a letter dated 21 March 1836 to the Danish-German astronomer, Heinrich Christian Schumacher [172], Gauss described the problem in terms of a railway connection between four towns in Germany. In a later restatement of the problem in 1941, in the book *What is Mathematics?* [123], the authors, Richard Courant and Herbert Robbins, call the problem the 'street network problem', suggesting its potential applications in the design of systems of roads.[21]

Despite these concrete formulations of the Steiner tree problem in the early literature, there is no evidence that the problem was ever studied as anything other than a problem in pure geometry before the 1950s. In the late 1950s, however, a genuine application emerged for the Euclidean Steiner tree problem. As Henry Pollak relates in [308], the Bell Telephone Company faced the issue that its tariff for private line service was written in terms of the length of the minimum network connecting the customers' stations (that is, the customer would not be charged for redundancy or inefficient interconnections in the network). In theory, a customer could insist that the Bell Service add a new terminal at a location that would reduce the cost of the minimum network (even if it was of no direct use to the customer), or at least that the customer be charged as though such a terminal existed. Hence, there was a real motivation for understanding how to compute the minimum network cost where such extra terminals (Steiner points) were allowed, which engaged the attention of a number of researchers working for or associated with Bell Laboratories.[22]

More recently, Durand et al. [145] have shown that an understanding of the local structure of minimum Steiner trees gives insight into the geometry of an electrical network made of resistive wires in which the averaged conductivity is maximised. In such networks, wires between junctions are straight, and each junction $(i)$ between two or more wires satisfies $\sum_j \mathbf{e}_{ij} = \mathbf{0}$, where $\mathbf{e}_{ij}$ are the outward-pointing unit vectors in the directions of the adjoining wires. Clearly, these are the same local conditions as those satisfied by edges and Steiner points in a Steiner tree. The extension of this problem to three dimensions has particular applications to estimating the conductivity of foams.

From the 1960s onwards, however, it has been variations on the Euclidean Steiner tree problem in the plane, rather than the original problem itself, that have found the most significant real-world applications. In 1966, Maurice Hanan published his highly influential paper on the Steiner tree problem in the rectilinear metric [189]. This paper establishes most of the basic properties of minimum Steiner trees where distance is measured using the rectilinear norm (also known as the Manhattan or $\ell_1$ norm), and correctly predicts the importance of this theory to the emerging area of 'printed circuit technology' (i.e., physical design of microchips). This problem and its applications are discussed in Chapter 3. More recently, a more general class of minimum trees, known as fixed orientation Steiner trees (which are the subject of Chapter 2), have been shown to have applications to the physical design of the next generation of microchips.

The Euclidean Steiner tree problem in the plane does, however, have some interesting links with other areas of pure and applied mathematics. Courant and Robbins [123], for example, suggested a relationship between Steiner trees and minimal surfaces. Minimal surfaces with constraints such as fixed boundaries or enclosing fixed volumes can be effectively realised using soap films, obtained by dipping an object into a soap or detergent solution. Courant and Robbins proposed an experiment in which two parallel glass or perspex plates are joined by three or more perpendicular bars, and the object is immersed in a soap solution and withdrawn. The resulting soap film forms a system of vertical planes between the plates and joining the fixed bars, the projection of which is a Steiner tree; see Figure 1.47. It is important to note, however, that this mechanism does not in general solve the Euclidean Steiner tree problem; the projection of the soap films will generally be a locally minimal network, but there is no guarantee of global minimality. A number of authors, including Miehle [282] and Dutta et al. [147], have built physical models to test this heuristic method. In the latter paper, the authors run a number of six-pin soap film experiments, where the pins are arranged as the vertices of a regular hexagon. They obtain a range of different Steiner trees, including but not restricted to the minimum one (which, by Theorem 1.19, consists of five sides of the regular hexagon); they also show that for thicker pins they obtain non-minimal spanning trees, or trees composed of Steiner components where the angles between edges at some terminals are less than $2\pi/3$.

A Euclidean Steiner tree for a fixed topology can also be viewed as an equilibrium system minimising directed forces on the edges or potential energy, which can be physically realised by a mechanism involving weights and strings (as in Exercise 1.4). This

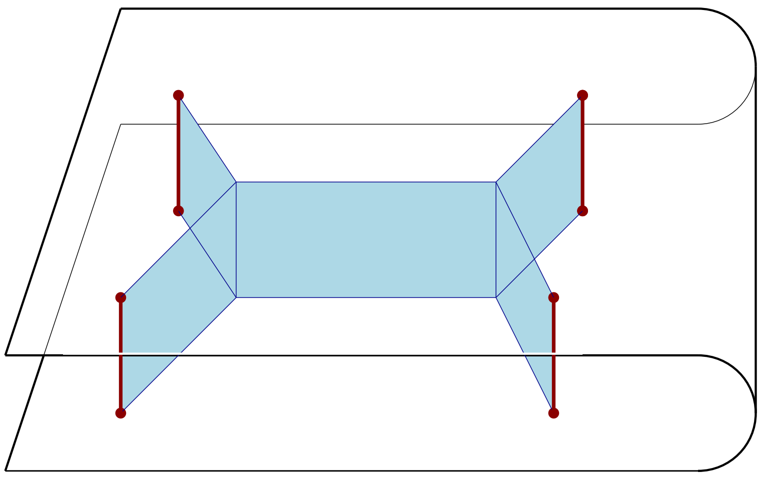


Figure 1.47: A soap film construction of a Steiner tree on four points.

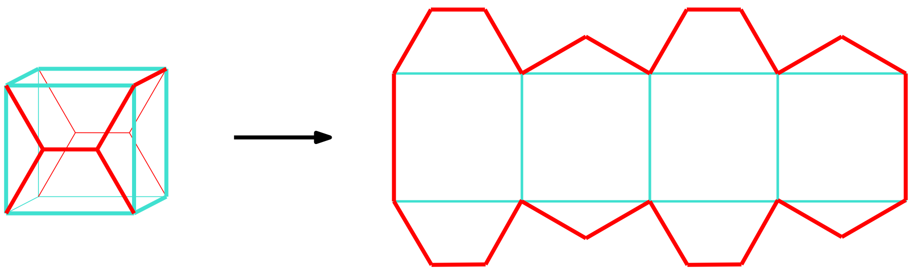


Figure 1.48: A minimum perimeter unfolding of a cube, obtained by finding a minimum Steiner tree (shown in red) on the surface of the cube interconnecting the eight vertices of the cube.

sort of model was proposed as early as 1811 by Gergonne [175], and has been physically implemented by Miehle [282].

Another problem that requires an understanding of the Euclidean Steiner tree problem in the plane for its solution, is that of constructing a minimum perimeter unfolding of a polyhedron. An *unfolding* (also known as a *development*) of a convex polyhedron is a connected plane figure obtained by cutting the surface of the polyhedron and unfolding it. The surface cuts can be made both along edges and across faces, and need not be restricted to straight cuts. Akiyama et al. [6] have shown that for any convex polyhedron an unfolding with minimum perimeter can be found by cutting along a minimum Steiner tree on the surface of the polyhedron interconnecting all the vertices. In Figure 1.48, for example, a minimum Steiner tree on the surface of a cube interconnecting the vertices of the cube is shown. By cutting along this tree the unfolding on the right is obtained. It is straightforward to show that this unfolding has minimum perimeter, since the perimeter is clearly twice the length of the tree interconnecting the vertices. Despite its three-dimensional setting, the problem of constructing a Steiner tree on the surface of a polyhedron is essentially a planar problem, and can be solved using the techniques in this chapter (see Exercise 1.23).

The Steiner tree problem has also been studied in a completely different context by Beardwood et al. in their well known 1959 paper [25], in which they show that the length of the shortest closed path through $n$ points in a bounded plane region of area $v$ is 'almost always' asymptotically proportional to $\sqrt{nv}$ as $n \to \infty$. The authors show that this asymptotic bound also applies to the Euclidean Steiner tree problem as the number of terminals in a bounded region increases.

### 1.7.2 Extensions to higher dimensions

A related Steiner tree problem that has useful applications is the Steiner tree problem in three (or higher) dimensional Euclidean space. This problem has been studied both as a natural generalisation of the planar problem and for its applications to areas such as phylogenetics ([78], [108], [62] and others) and to the structure and folding of proteins [352].

An interest in Steiner trees in higher dimensions dates back to the work of Bopp [36] in 1879 and to the first modern treatment of the Steiner tree problem by Jarník and Kössler [225] in 1934.[23] The formal statement of the Euclidean problem is essentially identical to that of the planar problem but in a more general setting.

EUCLIDEAN STEINER TREE PROBLEM IN $d$-DIMENSIONAL SPACE
**Given**: A set of points $N = \{t_1, \ldots, t_n\}$ lying in $d$-dimensional space (for $d \geq 2$).
**Find**: A geometric network $T = (V(T), E(T))$, such that $N \subseteq V(T)$, and such that $|T| := \sum_{e \in E(T)} |e|$ is minimised.

As in the planar case, a network solving the Euclidean Steiner tree problem in $d$-dimensional space is a tree, referred to as a *minimum Steiner tree*. All the other associated definitions including terminals, Steiner points, Steiner tree, meeting angles and full components carry across to this more general setting in a natural way.

In a $d$-dimensional Steiner tree the geometric properties of a Steiner point are remarkably similar to those in the planar case. Note first that any two edges of a Steiner tree incident to a single vertex are coplanar, and hence by the same argument as in the planar case the angle between them is at least $2\pi/3$. From this, it is straightforward to prove the following result (see Exercise 1.24).

**Theorem 1.40** *Let $s$ be a Steiner point of a $d$-dimensional Steiner tree $T$. Then $s$ has degree $3$ in $T$, each meeting angle at $s$ is $2\pi/3$, and the three edges incident with $s$ are coplanar.*

However, despite the local similarities with the planar case expressed in Theorem 1.40, the construction of a Steiner tree (for a given terminal set and given Steiner topology) is much harder in $d$-space than in the plane. The difficulty arises from the fact that although the edges at each Steiner point are coplanar, the plane those edges lie in is not known in advance if there are more than three terminals. This means that the Melzak-Hwang algorithm cannot be applied here, because there is no longer a unique equilateral point (or pair of equilateral points) associated with a cherry, but rather a continuous range of candidate equilateral points lying on a circle in $d$-space. In fact, it has been shown that for $d \geq 3$ and $n \geq 4$ (where $n$ is the number of terminals) the locations of the Steiner points cannot be exactly computed (in radicals), as the solution involves finding roots of a polynomial of degree $8$ [354, 279].[24]

A consequence of this is that numerical approaches are required for solving the Euclidean Steiner tree problem in $d$-dimensional space, $d \geq 3$. The best-known method is the iterative algorithm of Smith [354], discussed in Section 1.2.2. Smith's method involves enumerating all full Steiner topologies on the given set of terminals, and for each such topology computing the unique relatively minimal tree via a length-decreasing iterative process that has been shown to have linear convergence. In each case the resulting relatively minimal tree may be degenerate (i.e., have some edges of length zero); if a zero-length edge occurs between two Steiner points, then the relatively minimal tree cannot be a minimum Steiner tree, and hence the topology can be discarded. The shortest surviving relatively minimal tree is the required minimum Steiner tree.

In practice, Smith's algorithm can only solve relatively small instances of the Steiner tree problem, due to the large number of topologies that need to be considered. Smith's algorithm uses a branch-and-bound strategy to search through all possible full topologies. Improvements to the algorithm have been proposed and implemented by Fampa and Anstreicher [151], essentially by obtaining better lower bounds on the lengths of the candidate Steiner trees to reduce branching in the branch-and-bound tree. The improved algorithm, however, can still only solve problems with up to 15 or 16 terminals. Another easily computed lower bound on the length of relatively minimal trees has been suggested by Brazil et al. [68], but this is unlikely to be tight enough to give much additional improvement in a branch-and-bound algorithm.

Finally, we briefly mention the Steiner ratio $\rho_d$ for a $d$-dimensional Euclidean space (where $\rho_d$ is the infimum of the ratio of the length of a minimum Steiner tree and an MST for any set of terminals). As in the planar case, the precise Steiner ratio is not known for any $d \geq 3$. However, for $d = 3$ the terminal sets with the smallest known Steiner ratio are the so-called *3-sausages*, which are formed by gluing together faces of regular $3$-simplices (i.e., regular tetrahedra) so that their vertices lie on a helix. Smith and Smith [355] conjecture that as the length of the 3-sausage increases, the ratio of the minimum Steiner tree to the MST monotonically decreases, and in the limit converges to $\rho_d$. This conjectured value of $\rho_d$ is $\approx 0.78419037$. The low Steiner ratios associated with 3-sausages have been exploited in the design of heuristics for the $3$-dimensional Steiner tree problem, such as the algorithm of Toppur and Smith [373].

## Exercises

**1.1.** If $T$ is a star network solving the Fermat-Torricelli problem for $N = \{a, b, c\}$, show that $T$ is the minimum length network interconnecting the points of $N$. [Hint: In particular, we need to show that for a minimum length network interconnecting the points of $N$: the network must be a tree; we can assume that any vertex of the network not in $N$ has degree $3$ or more; and there can be at most one such vertex.]

**1.2.** Show that if $\triangle abc$ has an angle that is $2\pi/3$ or greater then the point that minimises the sum of distances to $a$, $b$ and $c$ coincides with the vertex with the largest angle. [Hint: Prove that i) the point cannot be inside $\triangle abc$, ii) the point cannot be outside $\triangle abc$, and iii) the point has to appear at a vertex if it is on the boundary of $\triangle abc$.]

**1.3.** Prove the following theorem which historically has been attributed to Napoleon (for a discussion see [400]). The centres of circumcircles $C_{ab}$, $C_{bc}$, $C_{ca}$ for a given triangle $\triangle abc$ form an equilateral triangle.

**1.4.** Show that the mechanical model for the Steiner tree problem for three given terminals, where the terminals correspond to the centres of three circular holes in a tabletop (see Figure 1.49), correctly identifies the location of the Steiner point for three given points.

**1.5.** Show that the edges of a minimum Steiner tree are straight line segments, and that they are embedded in such a way that their interior does not intersect any other vertex or edge of the minimum Steiner tree.

**1.6.** Show that a minimum Steiner tree with $n$ terminals has at most $n - 2$ Steiner points and at most $2n - 3$ edges. Hint: Suppose that the tree has $k$ Steiner points. Use the fact that a tree with $n + k$ vertices has $n + k - 1$ edges, and that all Steiner points have degree 3.

**1.7.** Show that a full Steiner topology with 4 or more terminals has at least two cherries, that is, pairs of terminals sharing a Steiner point.

**1.8.** Consider a Euclidean full minimum Steiner tree $T$. Show that for any edge $uv$ in $T$, there exists a merging order of the cherries of the topology, such that the final Simpson line overlaps with $e$.

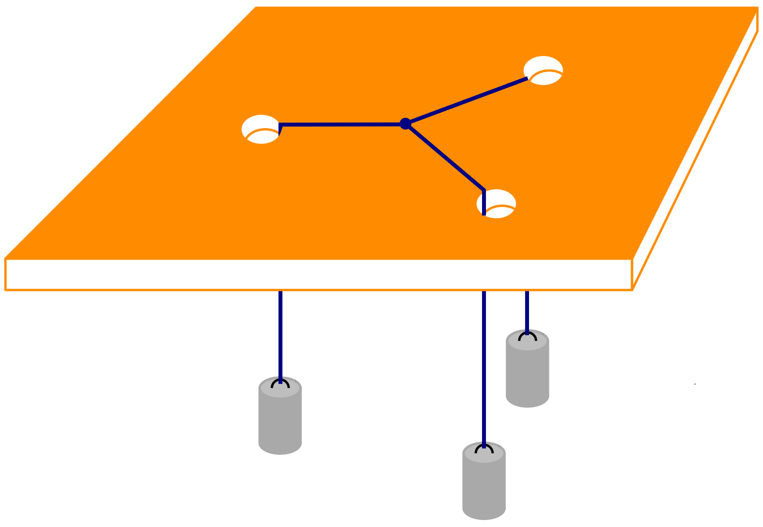


Figure 1.49: The mechanical model for the Steiner tree problem. A tabletop contains three circular holes, whose centres correspond to the locations of three terminals. Three strings with equal lengths, each with equal weights attached, pass through the holes and are tied together above the tabletop in a single knot. Assuming the system is frictionless, we claim that the resting position of the knot corresponds to the position of the Steiner point.

**1.9.** Show that if the coordinates of the terminals $N$ are given as *rational numbers*, then the coordinates of the equilateral points and Steiner points of a full Steiner tree for $N$ can be written in the form $\alpha + \beta\sqrt{3}$, where $\alpha$ and $\beta$ are rational numbers.

**1.10.** Show that $1/2$ is a lower bound for the Steiner ratio $\rho$. [Hint: Show that the distance traversed by an outer walk of a Steiner tree is greater than or equal to the length of a minimum spanning tree on the same set of terminals.]

**1.11.** Show that the Steiner ratio for $n$ terminals where $n = 3$ is $\sqrt{3}/2$. [Hint: Let $a, b, c$ be the three terminals, let $s$ be the Steiner point, and assume that $bc$ is the longest edge of triangle $\triangle abc$. Show that there exist points $b'$ and $c'$ on the line segments $bs$ and $cs$ respectively such that $\triangle ab'c'$ is an equilateral triangle.]

**1.12.** Consider removing a set $e_1, e_2, \ldots, e_k$ of distinct edges from a minimum Steiner tree. For each of the remaining components containing at least one terminal, choose one (arbitrary) terminal from this component. Let $N_R$ be the set of chosen terminals. Construct a minimum spanning tree (MST) over $N_R$ using BSD$(t_i, t_j)$ (defined in Section 1.3.2) as the distance between a pair of terminals $t_i, t_j \in N_R$. Let $L$ be the length of this MST. Show that $L \geq \sum_{l=1}^{k} |e_l|$.

**1.13.** Prove Lemma 1.13, the lune property. [Hint: The argument is similar to the proof of Lemma 1.11.]

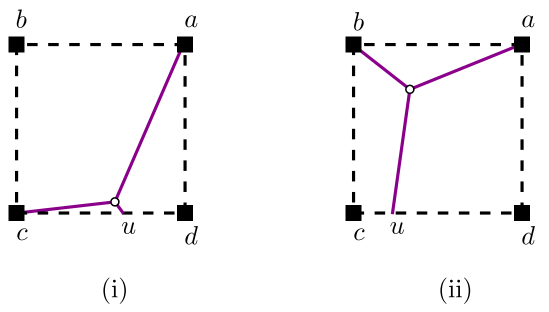


Figure 1.50: Two candidate topologies for the part of $T$ corresponding to a leaf of $G(T)$.

**1.14.** Let $s_1$, $s_2$ be adjacent Steiner points in a minimum Steiner tree $T$. Let $e_1$ and $e_2$ be two edges of $T$ incident with $s_1$ and $s_2$, respectively, and both lying on the same side of the line extending $s_1s_2$. Show, using Lemma 1.15, that if $|e_1| \leq |e_2|$ then $|s_1s_2| \geq (\sqrt{3}-1)|e_1|/2$.

**1.15.** In the proof of Theorem 1.17, calculate the length $|T_v|$ in terms of $R := \mathbf{d}(V_1', V_2'), r := \mathbf{d}(V_1, V_1')$ and $\{d_1, \ldots, d_n\}$.

**1.16.** Let $T$ denote a full component on $m$ terminals of a minimum Steiner tree $T^*$ for a rectangular lattice. Assume $\mathbf{e}(T) \leq 0$ and any connected graph on less than $m$ terminals has excess at least 0. Define the graph $G(T)$ to be the graph whose vertex set is $S$, two components in $S$ being adjacent in $G(T)$ if they both contain parts of the same edge of $T$ or if they both contain edges adjacent to a Steiner point on the boundary of a square. Without using Theorem 1.23, show that the part of $T$ occurring in a leaf of $G(T)$ cannot have either of the two topologies shown in Figure 1.50.

**1.17.** Show that there exists a full minimum Steiner tree on five terminals in the Euclidean plane that is not fulsome. [Hint: You may use the fact (which follows easily from [59]) that a Euclidean minimum Steiner tree on the set of terminals $\{(0,0)$, $(1,0)$, $(0,1)$, $(1,1)$, $(0,2)\}$ has length $2+\sqrt{3}$.]

**1.18.** Show that any configuration with centre $s$ that is part of a Steiner tree with Steiner point $s$ must be a Steiner configuration.

**1.19.** Prove, carefully, Theorem 1.26 for the case where $m > 3$.

**1.20.** Prove Lemma 1.30. [Hint: Suppose one of the supporting lines, say $L_2$, supports $\mathcal{C}$ at a point $x$ that is strictly between $L_1$ and $L_0$. Consider the line $L_y$ parallel to $L_2$, supporting $s+\mathcal{C}$ at $y$, the antipodal point to $x$ on $s+\mathcal{C}$. By comparing $L_3$ to $L_y$ show that if $L_2$ and $L_3$ intersect on the line $L$ then $|sw_2| \neq |sw_3|$.]

**1.21.** Prove Lemma 1.36. [Hint: Choose $s'$ on the interior of $sx_1$ or $sx_3$ such that there exist points $a \in sx_2$ and $b \in sx_4$ for which $sas'b$ is a parallelogram.]

**1.22.** Prove Lemma 1.38. [Hint: Argue by contradiction, and apply Lemma 1.37.]

**1.23.** Find a minimum perimeter unfolding for a regular tetrahedron.

**1.24.** Show that if every meeting angle at a vertex $s$ of a Steiner tree in $d$-dimensional Euclidean space is at least $2\pi/3$, then $s$ has degree at most $3$. Hence, prove Theorem 1.40.

## Notes

$^1$The problem was suggested by Pierre de Fermat (1601–1665) in his celebrated essay on minima and maxima [153]: “Let he who does not approve of my method attempt the solution to the following problem: Given three points in the plane, find a fourth point such that the sum of its distances to the three given points is a minimum”. The earliest known solution to Fermat’s problem was a geometric construction due to the Italian physicist and mathematician Evangelista Torricelli (1608–1647) [374]. For a comprehensive study of the problem, see Kupitz and Martini [242].

$^2$The more usual term for this point is *Fermat point* or *Fermat-Torricelli point*, and for the more general case with $n$ points minimising their distances to a single point, the point sought is also called the *Steiner-Weber point*. In this book we adopt the term Steiner point for all these cases, in order to emphasise the unity of this problem with the more general Steiner tree problems we discuss later. Essentially we are viewing the Fermat-Torricelli problem as the simplest non-trivial case of the Steiner tree problem.

$^3$This elegant construction is often attributed to Hofmann [202], but according to Honsberger [203, pp. 22–34], the Hungarian mathematician Tibor Gallai (1912–1992) discovered it independently. Some earlier attributions are mentioned in Kupitz and Martini [242], who name it the ”rotation proof”. Presentations of the construction (and some generalisations) can be found in [125, 130, 186, 203, 224, 241].

$^4$Bonaventura Cavalieri (1598–1647), an Italian mathematician and Jesuate, argued that the segments connecting the given points with the Fermat-Torricelli point meet at $120^\circ$ angles. The fact that the Simpson lines, which connect the third corner of the equilateral triangle to the opposite given point, also meet at the Fermat-Torricelli point was shown in 1750 by Thomas Simpson (1710–1761). In 1834, Heinen [192] proved that the length of the Simpson lines are equal to the sum of distances from the given points to the Fermat-Torricelli point.

$^5$The earliest known statement and analysis of the Euclidean Steiner tree problem was by the French mathematician and logician Joseph Diaz Gergonne (1771–1859). In 1810 Gergonne established his own mathematics journal entitled the *Annales de mathématiques pures et appliquées* but more generally known as the *Annales de Gergonne*. In the first volume of the journal from 1810/11 [175], Gergonne states the following problem: ”A number of cities are located at known locations on a plane; the problem is to link them together by a system of canals whose total length is as small as possible”. It is clear from the analysis that this is indeed the Euclidean Steiner tree problem, and Gergonne even proposes a method for solving the problem that basically is identical to Melzak’s algorithm from 1961 [280]. During the 19th century, three other statements of the problem were given by: Carl Friedrich Gauss (1777–1855) in a letter to Schumacher from March 19, 1836; Karl Bopp (1856–1905) in his dissertation [36] from 1879; and Eduard Hoffmann (1858–1923) in a celebration essay [201] from 1890. The first modern treatment of the problem was given by Vojtěch Jarník and Miloš Kössler in 1934 [225]. For a detailed account on the history of the Euclidean Steiner tree problem, see [47].

[6]Minimum Steiner trees are more typically referred to as ‘Steiner minimum trees’ or ‘Steiner minimal trees’ in the literature. The main reason for this appears to be to give them an acronym ‘SMT’ that is different from that of a minimum spanning tree ‘MST’. Mathematically, however, the term ‘Steiner minimum tree’ does not make sense (what is a ‘minimum tree’ and how can it be Steiner?)‘and hence in this book we avoid it. In some of the early literature terminals were also referred to as ‘regular points’, but ‘terminal’ seems preferable as the word ‘regular’ is ambiguous and heavily used in other areas of mathematics.

[7]We should note that this definition of a Steiner tree is different from the one in Gilbert and Pollak [180] and Hwang, Richards and Winter [213], but it is more operational.

[8]This recursion was first given by Bopp [36] in 1879. Gilbert and Pollak [180] gave both the recursion and the formula; they credit Riordan [327] with solving the recursion. The proof that any Steiner topology can be realised by some minimum Steiner tree was first given by Hwang et al. [215].

[9]The exponential but finite construction of Melzak [280] was given in 1961. Melzak’s algorithm was for a long time believed to be the first finite-time algorithm for solving the problem. More recently it has been discovered that Gergonne in 1810/11 [175] presented a solution method that basically is identical to the one suggested by Melzak 150 years later [47]. Hwang [212] improved the running time of Melzak’s original exponential algorithm to linear time. A number of implementation details were presented by Smith [354].

[10]The Euclidean Steiner tree problem in the plane does, however, have a polynomial-time approximation scheme [16]. This means that for any fixed $\epsilon > 0$ there exists a polynomial-time algorithm for computing a Steiner tree of length at most a factor $1 + \epsilon$ from optimum.

[11]Here we give a brief history of the Steiner ratio conjecture, based largely on the 2012 account by Ivanov and Tuzhilin [222], and their update in 2014 [223]. In 1992 a paper by Du and Hwang [136] presented a proof of the Steiner ratio conjecture. This proof, though generally accepted throughout most of the Steiner tree community, also caused some concern as the argument had a certain lack of precision and formality making it difficult to verify. The first paper to seriously question the proof in [136] was that of Yue [428], in 2000; however, the details of Yue’s criticism of the proof were unconvincing, and a proposed alternative proof was flawed. In 2001-2003 a number of mathematicians, including Ivanov, Tuzhilin, Morgan, Bern, Ceislik, Graham and others, discussed Du and Hwang’s proof via e-mail in some detail. Two major gaps were identified in the proof: one concerned the continuity of the inner spanning tree length and the other the reduction to full Steiner topologies. While some discussion of these gaps (by Ivanov and Tuzhilin) appeared in the Russian language literature at the time it was not until 2008 that these gaps were discussed in the English language mathematics literature, first by Ivanov and Tuzhilin [221], and shortly afterwards by DeWet [129] and Innami et al. [220]. The general consensus appears to be that there is no obvious way of fixing these gaps in the proof, hence the Steiner ratio conjecture remains open.

[12]This invariance property for the minimal Steiner hull was first stated 10 years earlier in Part 1, Theorem 1.5 of the book [213] by Hwang et al. However, Winter [407] gives a simple counter-example to show that a key assumption in the ‘proof’ of this result in [213], that any pair of consecutive legal replacements can be directly reversed, is incorrect.

[13]The terminology for a lune, which originated with Gilbert and Pollak [180] and is now standard in the Steiner tree literature, is inconsistent with the standard geometrical terminology. In planar geometry, the term ‘lune’ usually refers to a crescent-shaped convex-concave region bounded by two circular arcs, or in other words, the relative complement of one disc in another (where they intersect but neither is a subset of the other). A ‘lune’, as defined here, is an example of a ‘lens’ in standard planar geometry.

[14]A slightly weaker version of Lemma 1.15, with the added assumption that two different full Steiner trees exist for $N$, was first proved by Pollak [308]. Du et al. [138] later simplified Pollak’s proof, and showed that the existence of a second full Steiner tree is not required. They also showed that it is possible

for a full Steiner tree of $N$ that is not acute to be a minimum Steiner tree (if no other full Steiner tree exists).

[15]Cockayne and Schiller [115] developed the first computer code to compute minimum Steiner trees in 1972. The computer program enumerated terminal subsets and full Steiner topologies for each subset, and used Melzak's algorithm [280] to compute a full Steiner tree for a given full Steiner topology. The constructed full Steiner trees were then combined to form a minimum Steiner tree. Improved implementations of the algorithm were suggested by Boyce and Seery [39] and Boyce [38]. None of these implementations were able to compute minimum Steiner trees for more than 10 terminals. Winter [404] suggested the GeoSteiner approach in 1985, allowing the computation of minimum Steiner trees with more than 20 terminals. Cockayne and Hewgill [112, 113] suggested a number of improvements to Winter's algorithm that allowed the computation of minimum Steiner trees with up to 50 terminals.

[16]This use of the term "fulsome" is unfortunate, as the usual English meaning of the word is 'excessive' or 'insincere'. However, since the publication of [213] the term has become standard.

[17]This terminology is taken from [361].

[18]Theorem 1.27 was first stated by Cockayne [110], but without a complete proof. Full proofs were given by Levy [254] and Alfaro et al. [7], before the rather more elegant and general proof given by Lawler and Morgan [243].

[19]The approach here is loosely based on that outlined in [80] for the more restricted problem where the Minkowski norm has a strictly convex and differentiable unit circle. More details on the proof of this restricted problem, based on the method of Lagrange Multipliers, have also appeared in [177].

[20]Such an edge $e$ is referred to elsewhere in the literature [356] as a *trombone wire*. The slide described here is an example of a *zero-shift*, which we study in more detail in Chapter 2.

[21]For more details on the early history of the Euclidean Steiner tree problem, see [47]. One of the few modern papers further exploring the potential of the Euclidean Steiner tree problem in the design of road networks is the work of Stückelberger et al. [360].

[22]Employees or associates of Bell Laboratories who made important early contributions to the Euclidean Steiner tree problem in the plane include Zdzislaw Melzak [280], Edgar Gilbert [179], Henry Pollak [180], Ronald Graham and Frank Hwang [185].

[23]Karl Bopp's publication on Steiner trees took the form of a dissertation entitled "On the shortest connection system for four points" [36]. As the title suggests, the work considers the problem of how to construct a minimum Euclidean Steiner tree on four terminals. Although most of the focus is on the planar case, the thesis concludes with some brief observations on the three-dimensional problem, and states (without proof) some of the fundamental properties of three-dimensional Euclidean Steiner trees. These properties were proved and generalised to $d$-dimensional space (for all integers $d \geq 2$) in 1934 by the Czech mathematicians Jarník and Kössler [225], who were almost certainly unaware of Bopp's earlier work. This paper fell into obscurity until it was eventually rediscovered in the 1970s. Higher dimensional Steiner trees were not studied again in the literature until the seminal paper of Gilbert and Pollak [180] in 1968, where there is a short discussion of the three-dimensional problem.

[24]This non-computability result was first given by Smith [354], however the counter-example derived in that paper is difficult to verify. A clearer, more elegant counter-example was given by St. Mehlhos [279], who was apparently unaware of Smith's paper. An easier method for constructing counter-examples was later devised by Rubinstein et al. [332].

# Chapter 2

# Fixed Orientation Steiner Tree Problems

In this chapter we look at the problem of designing a network interconnecting a given set of points in the Euclidean plane, where each edge of the network is composed of straight line segments restricted to a fixed finite set of orientations, known as legal orientations. Associated with each of the legal orientations is a weight, and the aim is to find the interconnection network of this type with the minimum weighted length.

This is known as the fixed orientation Steiner tree problem. It has important applications in chip design, where millions of nets need to be routed on a (small) number of chip layers. On each routing layer, all wires generally use the same orientation in order to make joint routing of multiple nets feasible. In optimising the routing, the design of each net is usually treated as a planar geometric optimisation problem with fixed orientations, where the cost of transition between layers is treated as negligible.

The most well-known examples of fixed orientation networks are the rectilinear (or Manhattan) networks, where edges in the network are composed of horizontal and vertical line segments. The interest in such networks stems, at least in part, from the fact that at present most chip design technologies use only two perpendicular routing orientations. This strong restriction to only two legal orientations results in Steiner trees that have very specific geometric properties that will be studied in Chapter 3.

Here, however, we focus on the general fixed orientation Steiner tree problem, particularly where there are at least three legal orientations available. In chip design, the increasing number of available routing layers has made the use of multiple orientations relevant in practice [87, 365]. The fixed orientation Steiner tree problem can also provide a convenient way to approximate solutions to the Steiner tree problem in arbitrary norms (such as those in Section 1.6). These applications are discussed in greater detail in Section 2.7 of this chapter.

## 2.1 Fixed orientation networks

In this section, we show that representing edges of a network by weighted line segments with fixed orientations (and weights) induces a metric on the cost of edges. Here we establish a few fundamental properties of such a metric before defining the associated Steiner tree problem.

### 2.1.1 Fixed orientation metrics

Assume that a finite set of weighted orientations (referred to as *legal orientations*) is given. The weight $w_i$ of each orientation is assumed to be a positive real number. A convenient way to represent a weighted orientation is by a symmetric pair of vectors $\mathbf{u}_i, -\mathbf{u}_i$ where a line extending either vector has the required orientation, and the magnitude of $\mathbf{u}_i$ is chosen such that $|\mathbf{u}_i| = w_i^{-1}$. For each such vector $\mathbf{u}_i$, we let $u_i$ denote the corresponding point in the Euclidean plane with position vector $\mathbf{u}_i$.

Associated with a set of legal orientations is a centrally symmetric polygon $\mathcal{C}$ defined as follows.

**Definitions [Legal orientations and the orientation polygon]**: Suppose we are given a set of $k$ *legal orientations* each with weight $w_i$, $i = 0, \ldots, k-1$; we denote each orientation by a pair of vectors $\mathbf{u}_i, \mathbf{u}_{i+k}$ with $\mathbf{u}_i = -\mathbf{u}_{i+k}$ and $|\mathbf{u}_i| = w_i^{-1}$. The corresponding points in the Euclidean plane $u_0, u_1, \ldots, u_{2k-1}$ form the vertices of centrally symmetric polygon $\mathcal{C}$, centreed at the origin $o$, which we call the *orientation polygon*. We will assume that the set of legal orientations and weights are chosen so that the orientation polygon $\mathcal{C}$ is convex.

Note that although the convexity condition in the above definition seems restrictive, we will later show that if there is a legal orientation not satisfying the convexity condition, then that orientation is never required in a minimal network, and hence can be ignored.

An example of an orientation polygon defined by a set of three legal orientations is given in Figure 2.1 (left).

The networks that we study in this chapter are built up from paths of line segments in legal directions, defined as follows.

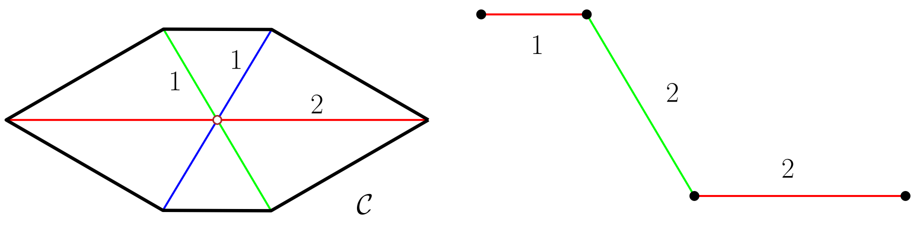


Figure 2.1: On the left is an example of an orientation polygon $\mathcal{C}$, defined by three legal orientations, each indicated with a different colour. On the right is an example of a fixed orientation path $\mathcal{P}$ with respect to $\mathcal{C}$. The length of $\mathcal{P}$ is the sum of the length of each line segment multiplied by its weight, which in this case is $(1\times 1/2)+(2\times 1)+(2\times 1/2) = 3.5$.

**Definition [Fixed orientation path]**: A path composed of line segments in legal directions between two given endpoints is referred to as a *fixed orientation path*; we calculate the length (or cost) of this path as the sum of the length of each line segment multiplied by its weight. For a given fixed orientation path, we refer to the endpoints of the path and the points shared by two adjacent line segments (with different orientations) as the *vertices* of the path.

An example of a fixed orientation path for a given set of legal directions is given in Figure 2.1 (right).

The above definition gives a natural way of measuring the distance between two points with respect to a given polygon $\mathcal{C}$, representing a set of fixed orientations. Given two points $p$ and $q$ we define the distance $|pq|_{\mathcal{C}}$ to be the minimum possible length of a fixed orientation path from $p$ to $q$. Similarly, given a path $\mathcal{P}_1 = v_1v_2\cdots v_m$ we define

$$|\mathcal{P}|_{\mathcal{C}} := \sum_{1}^{m-1} |v_iv_{i+1}|_{\mathcal{C}}.$$

If the orientation of the line through $p$ and $q$ coincides with a legal orientation, then the minimum fixed orientation path from $p$ to $q$ is simply the straight line segment $pq$, and is clearly unique. On the other hand, if the orientation of the line through $p$ and $q$ does not coincide with a legal orientation, then the minimum fixed orientation path from $p$ to $q$ contains at least one intermediate vertex. These observations motivate the following definitions.

**Definitions [Straight edges and bent edges]**: For two given points $p$ and $q$, if the line $\overline{pq}$ through $p$ and $q$ coincides with a legal orientation, then we say that the minimum fixed orientation path $pq$ (from $p$ to $q$) is a *straight edge*. If the orientation of $\overline{pq}$ does not coincide with a legal orientation, then the minimum fixed orientation path between $p$ and $q$ is called a *bent edge*, and the intermediate vertices of a bent edge are called *corner points*.

Note that a minimum path corresponding to a bent edge is composed of multiple legal line segments. We will show below (Theorem 2.1 to Corollary 2.4) that if we consider a line segment not with legal orientation; then any zigzag path between the endpoints consisting of legal line segments having the immediately preceding and succeeding legal orientations constitutes a shortest path; Figure 2.1 (right) is an example of such a shortest path. In fact, a bent edge can be assumed to consist of at most two line segments (having legal orientations) and a single corner point.

The above definitions and the following three theorems are largely based on the results of Widmayer et al. [402, 403], who proved equivalent results for the unweighted case.[25]

**Theorem 2.1** *There exists a shortest fixed orientation path between any two points composed of at most two line segments (and hence using at most two directions).*

**Proof.** Let the set of legal directions be determined by a convex centrally symmetric polygon $\mathcal{C}$, and suppose $\mathcal{P}$ is a fixed orientation path from $p$ to $q$ composed of at least three line segments in legal directions. There exists a subpath of $\mathcal{P}$, say $\mathcal{P}_1 = v_1v_2v_3v_4$, such that $v_1, v_2, v_3$ and $v_4$ are vertices of $\mathcal{P}$. Two examples of such subpaths are illustrated in Figure 2.2.

We will show that either $\mathcal{P}_1$ is not a minimal path between $v_1$ and $v_4$, or that there exists another fixed orientation path from $v_1$ to $v_4$ of the same length composed of at most two line segments.

Let $L_{1,2}$ be the line extending $v_1v_2$ and let $L_{3,4}$ be the line extending $v_3v_4$. Consider a perturbation of $\mathcal{P}_1$ to a path $\mathcal{P}_1'$ obtained by moving $v_2$ to a point $v_2'$ on $L_{1,2}$ and $v_3$ to a point $v_3'$ on $L_{3,4}$ such that $v_2'v_3' \parallel v_2v_3$. Here the location of $v_3'$ is determined by that of $v_2'$, so we can think of the movement of these points as being controlled by a single parameter: the distance $v_2$ moves along $L_{1,2}$. This is clearly a reversible perturbation; hence, there exists a direction of movement of the two points such that the instantaneous change in $|\mathcal{P}_1|_{\mathcal{C}}$ is either negative or zero.

If the instantaneous change in $|\mathcal{P}_1|_{\mathcal{C}}$ is negative, then $\mathcal{P}_1$ is not minimal. If, on the other hand, the change is zero, we note that the rate of change of the length of each line segment of $\mathcal{P}_1$ is constant, and hence the change in $|\mathcal{P}_1|_{\mathcal{C}}$ continues to be zero as we move

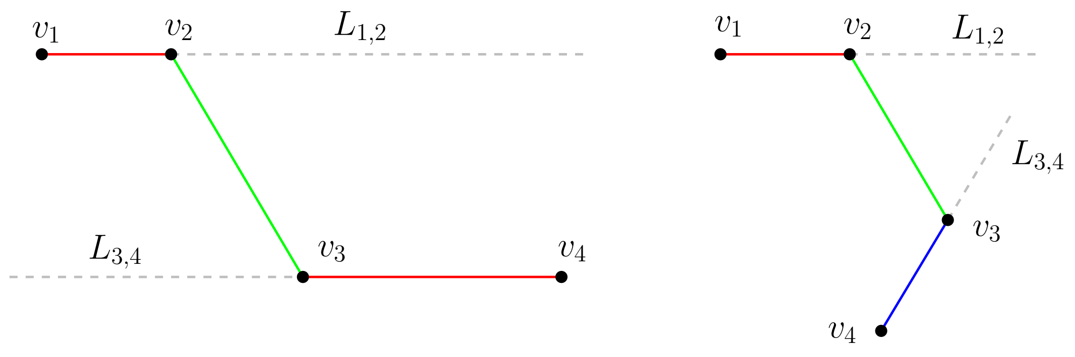


Figure 2.2: Two examples of subpaths composed of three line segments. In each case there exists a perturbation moving $v_2$ towards $v_1$ that does not change the length of the path on the left and strictly decreases the length of the path on the right.

$v_2$ along $L_{1,2}$. It follows that we can move $v_2$ along $L_{1,2}$ until either $v_2'$ coincides with $v_1$ or $v_3'$ coincides with $v_4$. The resulting path has the same length as $\mathcal{P}_1$ but is composed of at most two line segments.

The statement of the theorem now follows. ∎

Furthermore, it can be shown that any convex fixed orientation path containing two or more corner points is strictly non-minimal (Exercise 2.1).

A similar argument to the proof of Theorem 2.1 can also be used to show that if we choose a set of legal orientations such that the orientation polygon $\mathcal{C}$ is not convex, then for any pair of points $p$ and $q$ and for any orientation corresponding to a vertex of $\mathcal{C}$ that is not an extreme point of the convex hull of $\mathcal{C}$ there exists a shortest fixed orientation path between $p$ and $q$ not using that orientation (Exercise 2.2). This means that the restriction on the set of legal orientations to a set such that $\mathcal{C}$ is convex is not a restriction in the context of path minimisation, since we only need to consider orientations corresponding to extreme points of the convex hull of $\mathcal{C}$.

**Theorem 2.2** *The fixed orientation measurement of distance induces a metric on* $\mathbb{R}^2$.

**Proof.** From the definition of distance as the minimum possible length of a fixed orientation path between the given pair of points, it is immediately clear that the triangle inequality holds; i.e., for any three points $p$, $q$ and $r$ we have $|pq|_\mathcal{C} \leq |pr|_\mathcal{C} + |rq|_\mathcal{C}$, where equality holds if and only if $r$ lies on a minimum length fixed orientation path from $p$ to $q$. Similarly, the other properties of a metric, positive definiteness and symmetry, also follow immediately from the definition. ∎

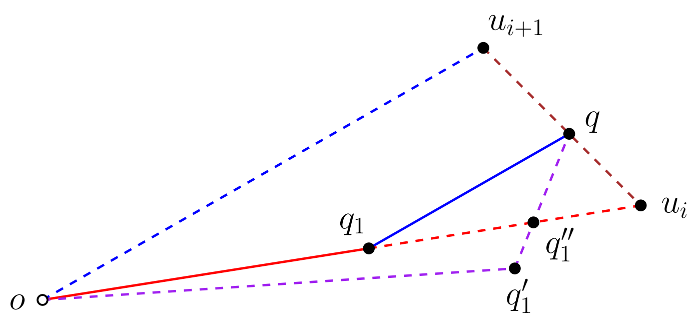


Figure 2.3: Construction for the proof of Theorem 2.3.

**Definition [Fixed orientation metric]**: The metric resulting from the fixed orientation measurement of distance is called a *fixed orientation metric*.[26]

**Theorem 2.3** *If $\mathcal{C}$ is a convex centrally symmetric polygon in the plane, centreed at the origin $o$, defining a fixed orientation metric, then the unit circle at $o$ for that metric is $\mathcal{C}$.*

**Proof.** The theorem is clearly true at the vertices of $\mathcal{C}$: for each vertex $u_i$ we have $|ou_i|_{\mathcal{C}} = w_i|u_i| = |u_i|^{-1}|u_i| = 1$. Let $u_i$ and $u_{i+1}$ be adjacent vertices of $\mathcal{C}$ and let $q$ be a point in the interior of $u_iu_{i+1}$. We will show that $|oq|_{\mathcal{C}} = 1$, from which the theorem follows.

By Theorem 2.1 there exists a minimum length fixed orientation path $\mathcal{P}_q$ between $o$ and $q$ composed of exactly two legal line segments. Since $\mathbf{u}_i$ and $\mathbf{u}_{i+1}$ are adjacent legal directions it follows that $\mathcal{P}_q$ intersects the interior of either $ou_i$ or $ou_{i+1}$; we assume, without loss of generality, that it intersects $ou_i$, as illustrated in Figure 2.3.

Note that there exists a unique fixed orientation path $\mathcal{P}_1$ between $o$ and $q$ composed of two legal line segments with directions $\mathbf{u}_i$ and $\mathbf{u}_{i+1}$, such that the interior vertex of $\mathcal{P}_1$, say $q_1$, lies in the interior of $ou_i$. We also note that $\triangle q_1qu_i \sim \triangle ou_{i+1}u_i$. It follows that $|q_1q|_{\mathcal{C}} = |q_1u_i|_{\mathcal{C}}$, and hence that $|\mathcal{P}_1|_{\mathcal{C}} = 1$.

It now remains to show that $\mathcal{P}_1$ is a minimum fixed orientation path. Let $\mathcal{P}_1'$ be any fixed orientation path between $o$ and $q$ that is composed of two legal line segments, that intersects the interior of $ou_i$, and that uses at least one direction other than $\mathbf{u}_i$ and $\mathbf{u}_{i+1}$. Since $\mathbf{u}_i$ and $\mathbf{u}_{i+1}$ are adjacent, the interior vertex of $\mathcal{P}_1'$, $q_1'$, does not lie in the interior of the cone from the origin induced by $\mathbf{u}_i$ and $\mathbf{u}_{i+1}$. It follows that $q_1'q$ intersects the interior of $q_1u_i$ at a point, say $q_1''$ (not necessarily different than $q_1'$). Applying the triangle inequality (from Theorem 2.2) twice, we obtain

$$\begin{aligned}|\mathcal{P}_1'|_\mathcal{C} &= |oq_1'|_\mathcal{C} + |q_1'q|_\mathcal{C}\\ &\geq |oq_1''|_\mathcal{C} + |q_1''q|_\mathcal{C}\\ &= |oq_1|_\mathcal{C} + |q_1q_1''|_\mathcal{C} + |q_1''q|_\mathcal{C}\\ &> |oq_1|_\mathcal{C} + |q_1q|_\mathcal{C} = |\mathcal{P}_1|_\mathcal{C}\end{aligned}$$

as required (where the final strict inequality follows from the proof of Theorem 2.2). ∎

An immediate corollary is as follows:

**Corollary 2.4** *Let $p$ and $q$ be points in $\mathbb{R}^2$ such that the orientation of the line $\overline{pq}$ through $p$ and $q$ does not coincide with a legal orientation. Then $\mathcal{P}$ is a shortest fixed orientation path from $p$ to $q$ if and only if $\mathcal{P}$ is a zigzag path from $p$ to $q$ consisting of line segments having the immediate preceding and succeeding legal orientations to that of $\overline{pq}$.*

### 2.1.2 The fixed orientation Steiner tree problem

In this section we consider the Steiner tree problem for fixed orientation networks, for a given set of weighted legal orientations. The preceding theorems show that this is equivalent to solving the Steiner tree problem in a Minkowski plane where the unit circle $\mathcal{C}$ is a convex centrally symmetric polygon. We could define the Steiner tree problem here in terms of networks embedded in this Minkowski plane, as in the definition in Section 1.6. However, in order to best exploit the geometry of these networks, it is more convenient to express the problem in terms of fixed orientation networks embedded in the Euclidean plane. By a *fixed orientation network* we mean one where each edge in the network is a fixed orientation path between its endpoints.

Formally, the definition of the problem is as follows:

FIXED ORIENTATION STEINER TREE PROBLEM IN THE PLANE
**Given**: A set of points $N = \{t_1, \ldots, t_n\}$ lying in the plane, and a convex centrally symmetric polygon $\mathcal{C}$.
**Find**: A fixed orientation network $T = (V(T), E(T))$, such that $N \subseteq V(T)$, and such that $\sum_{e \in E(T)} |e|_\mathcal{C}$ is minimised.

As in Chapter 1, a solution to this problem is a tree, referred to as a *minimum fixed orientation Steiner tree*. The vertices of such a tree not in $N$ are called *Steiner points*. Furthermore, by minimality the edges of such a tree are minimum fixed orientation

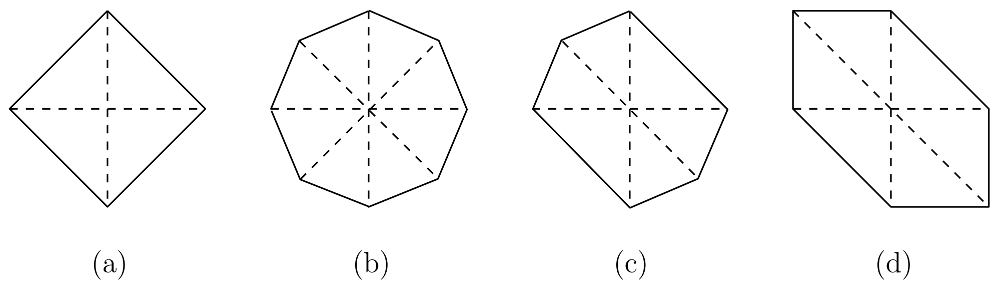


Figure 2.4: Examples of polygonal unit circles. (a) Rectilinear (two perpendicular orientations). (b) Uniform orientations ($\lambda = 4$). (c) Non-uniform orientations (with three unweighted orientations). (d) General weighted non-uniform orientations.

paths between their endpoints, and hence they are either straight line segments with legal orientation, or they satisfy the properties of Corollary 2.4.

For the fixed orientation Steiner tree problem we again have associated concepts of *topology*, *Steiner topology* and *Steiner trees*. These are defined in exactly the same way as in Chapter 1: in particular, for a given polygonal unit circle $\mathcal{C}$, the *topology* of a fixed orientation network is its underlying graph structure; a *Steiner topology* is a topology that can be realised by some non-degenerate minimum fixed orientation Steiner tree; and a *Steiner tree* is a non-degenerate relatively minimal fixed orientation tree for a Steiner topology.

The best known example of a Minkowski plane with polygonal unit circle is the rectilinear plane, where there are two fixed orientations: horizontal and vertical (Figure 2.4(a)) with equal weights. Special properties of the rectilinear Steiner tree problem are discussed in Chapter 3. A useful generalisation of the rectilinear plane is as follows:

**Definitions [$\lambda$-geometry]**: The $\lambda$-*geometry plane* is defined to be a Minkowski plane in which the unit circle $\mathcal{C}$ is a regular $2\lambda$-gon for some integer $\lambda \geq 2$ (as illustrated in Figure 2.4(b)). The corresponding metric is known as a *uniform orientation* metric [340]. A minimum Steiner tree in a $\lambda$-geometry plane is referred to as a *minimum* $\lambda$-*geometry Steiner tree*. We denote by $\omega := \pi/\lambda$ the smallest possible angle between two distinct legal orientations in a given $\lambda$-geometry.

Unit circles for some general fixed orientation metrics are shown in Figure 2.4. Figure 2.4(c) shows a polygon whose vertices lie on a Euclidean unit circle, meaning that the weight associated with each legal orientation is $1$. Finally, Figure 2.4(d) illustrates a case where the orientations are non-uniform and have different weights. By convention, we always assume that one of the legal orientations is horizontal.

Despite our Euclidean point of view in this chapter, where fixed orientation trees are viewed as trees embedded in the Euclidean plane composed of weighted line segments in legal directions, it is nevertheless convenient to express the length of an edge in a fixed orientation Steiner tree, or indeed the whole tree itself, in terms of the Minkowski norm (for the given polygonal unit ball), $\|\cdot\|$. This will be our convention throughout the remainder of this chapter. This has a number of advantages, including keeping the notation consistant with that used in Section 1.6.

In the next two sections we examine geometric properties of the Steiner points and full components of a fixed orientation Steiner tree.

## 2.2 Local properties of Steiner points

We begin our study of fixed orientation Steiner trees with local properties, and in particular the geometry of these trees around an individual Steiner point. Here we present bounds on the degrees and angles for Steiner points, first for the uniform orientation cases and then for general fixed orientation metrics. The relatively straightforward theory for the uniform orientation cases can be viewed as a warm-up for the more complex theory for fixed orientation metrics. For the more general case we make use of the *centroid property* developed in Section 1.6.1.

Although we focus on Steiner points, degree and angle bounds can also be obtained for terminals, but from an algorithmic point of view, properties related to Steiner points are substantially more interesting.

### 2.2.1 Steiner points for uniform orientation metrics

One of the most important fundamental results in the Euclidean case is that in a Euclidean Steiner tree no angle at a vertex (i.e., a terminal or Steiner point) is less than $2\pi/3$. It immediately follows from this that every Steiner point has degree $3$, and that the angles at any Steiner point are all exactly $2\pi/3$.

Clearly, these results do not, in general, hold for minimum $\lambda$-geometry Steiner trees, since all angles in such trees must be multiples of $\omega = \pi/\lambda$. We will, however, show that results very close to the Euclidean result do hold; for example, the angles at a Steiner point do not differ from $2\pi/3$ by more than $\omega$, and for most values of $\lambda$ all Steiner points are of degree $3$ .

We first consider the meeting angles at Steiner points in $\lambda$-geometry.

As discussed earlier, we assume that all given minimum $\lambda$-geometry Steiner trees are embedded in the Euclidean plane. Hence, we have an embedding of each of the edges as a minimum fixed orientation path. This means that each meeting angle at a vertex is an

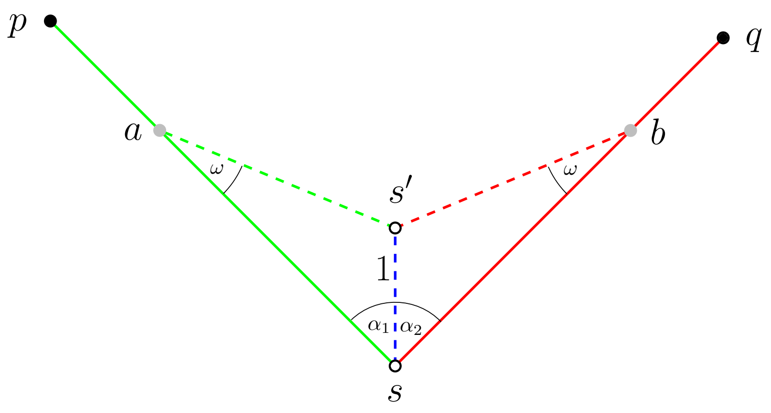


Figure 2.5: A non-minimal meeting angle, illustrated for the case where $\lambda = 8$.

angle between line segments in legal orientations.

**Theorem 2.5** *In a minimum $\lambda$-geometry Steiner tree, the minimum meeting angle at any Steiner point is at least $\lceil 2\lambda/3 - 1 \rceil \omega$, while the maximum meeting angle is at most $\lfloor 2\lambda/3 + 1 \rfloor \omega$.*[27]

**Proof.** The theorem states that the minimum possible angle at a Steiner point is the largest multiple of $\omega$ that is strictly less than $2\lambda/3$ and, similarly, that the maximum possible angle is the smallest multiple of $\omega$ that is strictly greater than $2\lambda/3$.

We first prove the minimum meeting angle condition. The argument is essentially by contradiction. First note that in any minimum $\lambda$-geometry Steiner tree the minimum meeting angle is strictly greater than $\pi/2 - 2\omega$ (Exercise 2.4). Let $T$ be a $\lambda$-tree interconnecting a given terminal set, and let $s$ be a vertex of $T$ with incident legal straight line segments $sp$ and $sq$ such that

$$\pi/2 - 2\omega < \angle psq < \lceil 2\lambda/3 - 1 \rceil \omega.$$

Let $s'$ be a point in the cone induced by the rays from $s$ extending $sp$ and $sq$ such that $|ss'| = 1$ and $ss'$ is in a legal orientation as close as possible to the bisector of the orientations of $sp$ and $sq$ (as illustrated in Figure 2.5).

By rescaling, if necessary, it follows that there exist points $a$ and $b$ on $sp$ and $sq$, respectively, such that $\angle sas' = \angle sbs' = \omega$. Note that $s'a$ and $s'b$ both have legal orientations. We will show that replacing $sa$ and $sb$ by the three line segments $s's$, $s'a$ and $s'b$ reduces the length of $T$, from which the minimum meeting angle condition follows.

Let $\angle ass' = \alpha_1$ and $\angle bss' = \alpha_2$ (as shown in Figure 2.5). By the angle conditions on $s$ we have

$$\pi/2 - 2\omega < \alpha_1 + \alpha_2 \leq 2\pi/3 - \omega.$$

Since $ss'$ is in a legal orientation as close as possible to the bisector of the orientations of $sp$ and $sq$ we have

$$\alpha_1 - \alpha_2 \leq \omega.$$

We want to show that $|s'a| + |s'b| + 1 < |sa| + |sb|$. By applying the sine rule on $\triangle ass'$ and $\triangle bss'$, and using standard trigonometric identities, we obtain:

$$\begin{aligned}
&\sin(\omega)(|sa| + |sb| - |s'a| - |s'b|) \\
&\qquad = \sin(\alpha_1 + \omega) + \sin(\alpha_2 + \omega) - \sin(\alpha_1) - \sin(\alpha_2) \\
&\qquad = 4 \sin\left(\frac{\omega}{2}\right) \cos\left(\frac{\alpha_1 - \alpha_2}{2}\right) \cos\left(\frac{\alpha_1 + \alpha_2 + \omega}{2}\right) \\
&\qquad > 4 \sin\left(\frac{\omega}{2}\right) \cos\left(\frac{\omega}{2}\right) \cos(\pi/3) = \sin(\omega)
\end{aligned}$$

as required. Note that the inequality in the above system makes use of both the upper and lower bounds on $\angle psq$.

We prove the maximum meeting angle condition by a similar construction. Let $s$ be a vertex of $T$ such that the minimum meeting angle condition holds but the maximum meeting angle condition does not; i.e., there are incident legal straight line segments $sp$ and $sq$ such that $\angle psq$ is the largest angle at $s$ and is strictly greater than $2\pi/3 + \omega$. Note that this immediately implies that $s$ has degree 3. By rescaling if necessary, let $s'$ be a point on the remaining straight line segment incident with $s$ such that $|ss'| = 1$, and again let $a$ and $b$ be the points on on $sp$ and $sq$, respectively, such that $\angle sas' = \angle sbs' = \omega$ (as illustrated in Figure 2.6).

As before, let $\angle ass' = \alpha_1$ and $\angle bss' = \alpha_2$. By the angle conditions on $s$ we have

$$\alpha_1 + \alpha_2 \leq 4\pi/3 - \omega.$$

Since the minimum meeting angle condition holds, it follows that $ss'$ is in a legal orientation as close as possible to the bisector of the orientations of $sp$ and $sq$; thus

$$\alpha_1 - \alpha_2 \leq \omega.$$

We want to show that $|sa| + |sb| + 1 > |s'a| + |s'b|$, and hence that $T$ is not minimal. A similar calculation to that in the minimal case shows that $\sin(\omega)(|s'a|+|s'b|-|sa|-|sb|) < \sin(\omega)$, as required.

It follows that the minimum and maximum meeting angle conditions must both hold in a minimum Steiner $\lambda$-tree. ∎

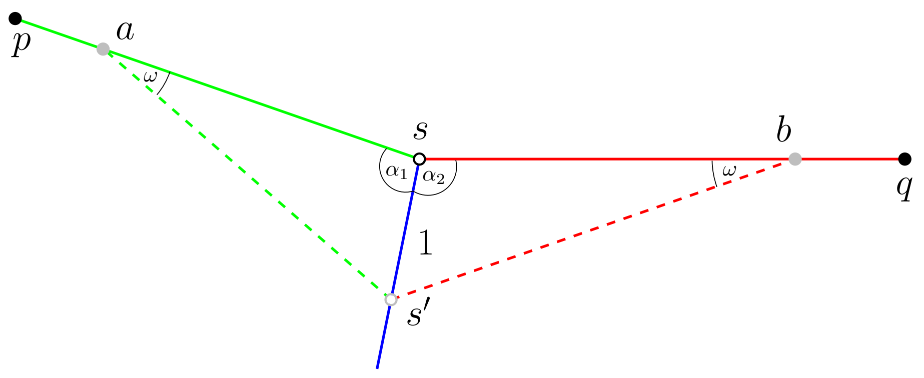


Figure 2.6: A non-minimal Steiner point, illustrated for the case where $\lambda = 9$.

In fact, a slightly stronger statement is possible for the minimum possible meeting angle at a Steiner point in a minimum $\lambda$-geometry Steiner tree (see Exercise 2.5).

Obviously, if the minimum angle $\lceil 2\lambda/3 - 1\rceil\omega$ is strictly greater than $\pi/2$, then the maximum Steiner point degree is $3$. Direct computation shows that degree $4$ Steiner points are only possible for $\lambda = 2, 3, 4$ and $6$. Thus we have the following:

**Corollary 2.6** *In $\lambda$-geometry, Steiner points have degree 3, except when $\lambda = 2, 3, 4$ and $6$ (where Steiner points with degree 4 exist).*

Note, however, that in the cases $\lambda = 3$ and $\lambda = 6$ a degree $4$ Steiner point can be split into a pair of degree $3$ Steiner points. Furthermore, it can be shown that in both of these cases $\lambda$-geometry Steiner points of degree $5$ or $6$ never occur (see Exercise 2.6).

If we think of edges as being composed of line segments in legal orientations embedded in the Euclidean plane, then for $\lambda = 3m$, a meeting angle can either be $2\pi/3 - \omega$, $2\pi/3$ or $2\pi/3 + \omega$, while for $\lambda \neq 3m$, only two meeting angles are possible.

## 2.2.2 Steiner points for general fixed orientations

We now generalise the above results by giving a complete characterisation of the distribution of meeting angles of Steiner points in minimum Steiner trees for a metric defined by a polygonal unit circle $\mathcal{C}$. By Theorems 1.34 and 1.39, we can assume that there are no Steiner points of degree $> 4$ and that any degree $4$ Steiner point in a Steiner tree is a cross and is adjacent to four terminals. Such points are algorithmically trivial to find and construct. Hence, in this section, and throughout most of the remainder of this chapter, we assume all Steiner points have degree $3$.

As we will see later in this chapter, the characterisation of meeting angles for degree $3$ Steiner points is a crucial step in developing fast algorithms for finding a fixed orientation Steiner tree for a given Steiner topology.

We first extend the concept of legal orientations to that of legal directions.

**Definition [Legal directions]**: Let $\mathbf{u}_l$, $l = 0, \ldots, 2\sigma - 1$ be the $2\sigma$ vectors that define the vertices of the unit circle $\mathcal{C}$ (in counter-clockwise order around the circle). These are unit vectors in the metric given by $\mathcal{C}$; that is, they have length 1 under metric $\| \cdot \|$. By the central symmetry of $\mathcal{C}$, it follows that $\mathbf{u}_l = -\mathbf{u}_{(l+\sigma) \bmod 2\sigma}$. Each unit vector $\mathbf{u}_l$ corresponds to a *legal direction*; that is, under this metric, any (oriented) legal line segment must use one of the $2\sigma$ unit vector directions. The *successor* of a unit vector $\mathbf{u}_l$ is the vector $\mathbf{u}_{l+1}$ where $\mathbf{u}_{l+1} := \mathbf{u}_0$ if $l = 2\sigma - 1$.

Throughout this chapter we will make frequent use of the assumption that the unit circle $\mathcal{C}$ is placed somewhere in the plane such that its centre coincides with a given point $s$. For such a fixed centre we can unambiguously refer to the *vertices* of $\mathcal{C}$, which are the endpoints of the unit vectors $\mathbf{u}_l$ for this fixed centre. The vertex that corresponds to the endpoint of a unit vector $\mathbf{u}_l$ is denoted by $u_l$ for $l = 0, \ldots, 2\sigma - 1$.

Recall that we refer to a star network centred at a Steiner point that is part of some full minimum Steiner tree as a *Steiner configuration*. In order to understand which sets of legal directions can occur in a Steiner configuration we make use of the centroid property established in Section 1.6 (Theorem 1.28). The aim is to develop a way of constructing all possible sets of supporting lines satisfying the centroid property, each of which corresponds to a feasible set of directions. We begin by formalising the notion of a direction set.

**Definitions [Maximal Steiner configuration, direction set]**: For any Steiner configuration of degree $3$ there is an associated set of legal directions, namely the legal directions used by all edges in the star (where directions are considered as oriented outward from the centre). A Steiner configuration $\mathcal{S}$ is said to be *maximal* if there exists no other Steiner configuration (for any set of terminals) that uses a strict superset of the legal directions used by $\mathcal{S}$. We define a *direction set* $\mathcal{D} = \{\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_k\}$ to be a set of legal directions used by a maximal Steiner configuration, listed in counter-clockwise order around the centre.[28]

The following definition will also prove useful:

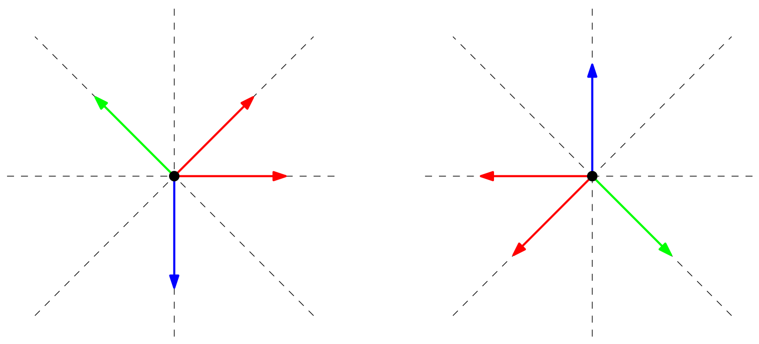


Figure 2.7: Complementary direction sets in $\lambda$-geometry for $\lambda = 4$.

**Definition [Complementary direction set]**: For a direction set $\mathcal{D}$ we define the *complementary* direction set as the direction set that is obtained by reversing all directions in $\mathcal{D}$ (Figure 2.7). By the central symmetry of $\mathcal{C}$, direction sets appear as *pairs* of complementary direction sets.

Lemma 2.7 below shows that for any direction set $\mathcal{D} = \{\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_k\}$ we have $k = 4, 5$ or 6.

**Lemma 2.7** *A direction set contains at least* $4$ *and at most* $6$ *distinct directions.*

**Proof.** The correctness of the upper bound is clear, since each edge uses at most two legal directions (by Corollary 2.4). We establish the lower bound using a continuity argument.

Suppose, contrary to the claim of the lemma, there exists a direction set $\mathcal{D}$ with only 3 directions. Let $S$ be a Steiner configuration with 3 leaves, $x_1, x_2, x_3$ and a Steiner point $s$, such that $S$ has direction set $\mathcal{D}$. By convexity, the choice of $s$ is unique, since if there were a second Steiner point $s'$ that gave a Steiner configuration, then every point in the line segment $ss'$ would also be the Steiner point of a minimum Steiner tree. This would mean that a larger direction set strictly containing $\mathcal{D}$ could be found by moving the Steiner point from $s$ a small distance towards $s'$.

For all points $p$ in the plane define the function $f(p) := \|x_1p\| + \|x_2p\| + \|x_3p\|$. Since $S$ is a Steiner configuration it follows that, for every $p$, $f(p) \geq f(s) = \|S\|$. For every $\varepsilon > 0$, let $B_\varepsilon$ be the open Euclidean ball with centre $s$ and radius $\varepsilon$. By the continuity of $f$ it follows that for each $\varepsilon$ there exists $\delta = \delta(\varepsilon) > 0$ such that $f(p) > f(s) + \delta$ for all $p \notin B_\varepsilon$. Clearly, we can also assume that $\delta < \varepsilon$.

Now choose $\varepsilon > 0$ sufficiently small such that perturbing one or both endpoints of each edge $x_is$ by at most $\varepsilon$ results in an edge in the normed plane that still uses a direction from $\mathcal{D}$ (for $i \in \{1, 2, 3\}$). Suppose we move leaf $x_1$ to $x_1'$ not on the line through $x_1s$ such that $0 < \|x_1x_1'\| < \delta(\varepsilon)$ and such that $S'$, the Steiner configuration for $x_1', x_2, x_3$, satisfies $\|S'\| \leq \|T\|$. Let $s'$ be a Steiner point for $S'$. Then

$$\begin{aligned} f(s') &\leq \|S'\| + \|x_1 x'_1\| \\ &< \|T\| + \delta. \end{aligned}$$

Hence, $s'$ lies in $B_\varepsilon$. By our choice of $\varepsilon$ it follows that $\mathcal{D}'$, the direction set of $S'$, contains $\mathcal{D}$ as a subset (since the endpoints of the three edges of $S$ have each been perturbed by at most $\varepsilon$). Since exactly one of the terminals of $S$ has been perturbed it is easy to see that the direction set of $S'$ is not the same as the direction set for $S$. Hence, $\mathcal{D}$ is a *strict* subset of $\mathcal{D}'$, contradicting the statement that $\mathcal{D}$ is a direction set. ∎

The above lemma shows that a maximal Steiner configuration always has at least one edge that contributes two legal directions to the corresponding direction set. We now use this observation to define a colouring scheme for edges in a maximal Steiner configuration, which we extend to directions in a direction set.

The colouring scheme is as follows. We colour one of the edges of a maximal Steiner configuration that contributes two legal directions to the directions set *red* and the other edges *green* and *blue*, respectively, in counter-clockwise order from the red edge. We extend these colour labels to the directions in the direction set.

**Definition [Coloured direction set]**: A direction set $\mathcal{D} = \{\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_k\}$ with coloured directions (red, green and blue) based on a colouring of edges in the maximal Steiner configuration is denoted a *coloured direction set*. Without loss of generality, we assume that a coloured direction set contains two red directions, $\mathbf{d}_1$ and $\mathbf{d}_2$; these are adjacent legal directions.

Note that for each given colour in a coloured direction set we have either one or two legal directions. This leads to the following definition.

**Definitions [Primary and secondary directions]**: When we have two directions in a direction set with the same colour, these are labelled as the (exclusively) *primary* and the (exclusively) *secondary* direction, respectively, in counter-clockwise order around the unit circle. When we have a single direction for a given colour, this direction can be labelled either primary or secondary.

It will sometimes be convenient to refer to all the primary (or secondary) line segments in a fixed orientation Steiner tree as constituting the *primary* (or respectively *secondary*) *material* in the tree.

Assume we fix a pair of adjacent red directions, which correspond to two unit vectors $\mathbf{u}_i$ and $\mathbf{u}_{i+1}$ of $\mathcal{C}$. How many coloured direction sets exist for this pair of red edges? In the proof of Theorem 2.9 below we show that there exists either one or two direction sets for each fixed pair of red directions. We then show that the collection of all maximal direction sets can be constructed in linear time.

**Notation:** For the remainder of this section we make use of some of the notation established in Chapter 1 (Section 1.6). In particular, given a unit circle $\mathcal{C}$ with centre $s$, and given a supporting line $L_1$, we denote by $L_0$ the line that is parallel to $L_1$ and contains $s$, and, if $d$ is the Euclidean distance between the parallel lines $L_1$ and $L_0$, then we denote by $L$ the line parallel to $L_1$ at distance $3d$ from $L_1$ (and at distance $2d$ from $L_0$).

Recall (from Lemma 1.29) that the conditions we require on two additional supporting lines $L_2$ and $L_3$ for the centroid property to hold (i.e., for $s$ to be the centroid of the triangle formed by $L_1, L_2$ and $L_3$) are that $L_2$ and $L_3$ intersect on $L$ and that the distances from $s$ of their respective intersections with $L_0$ are equal.

**Lemma 2.8** *Given a polygonal unit circle $\mathcal{C}$ with centre $s$ and a supporting line $L_1$, let $L_0$ and $L$ be defined as above.*

*(a) If $L_0$ does* not *intersect a vertex of $\mathcal{C}$, then there exists exactly one pair of supporting lines $L_2$ and $L_3$, such that $L_1$, $L_2$ and $L_3$ have the centroid property;*

*(b) If $L_0$ does intersect a vertex of $\mathcal{C}$, then either*

- *there exists exactly* one pair *of supporting lines $L_2$ and $L_3$, such that $L_1$, $L_2$ and $L_3$ have the centroid property, or*
- *there exists an infinite set of supporting lines $L_2$ and $L_3$ such that $L_1$, $L_2$ and $L_3$ have the centroid property, and $L_2$ and $L_3$ both support $\mathcal{C}$ at vertices of $\mathcal{C}$.*

**Proof.** Statement (a) has been proved in Lemma 1.31 of Chapter 1.

For statement (b), note that if $L_0$ intersects a vertex of $\mathcal{C}$, then by the central symmetry of $\mathcal{C}$ it intersects two vertices, say $u_j$ and $u_k$. The predecessor of $u_j$ (respectively $u_k$) on $\mathcal{C}$ is $u_{j-1}$ (respectively $u_{k-1}$) and the successor is $u_{j+1}$ (respectively $u_{k+1}$).

Let $z_j^-$ be the intersection of the line through $u_j$ and $u_{j+1}$ with $L$, and let $z_j^+$ be the intersection of the line through $u_{j-1}$ and $u_j$ with $L$; define $z_k^-$ and $z_k^+$ similarly (see Figure 2.8).

Now we distinguish between three cases, depending on the order (from left to right) in which the points $z_j^-$ and $z_k^+$ appear on $L$:

1. $z_j^- < z_k^+$: In this case any point $z$ in the interval $[z_j^-, z_k^+]$ defines a pair of supporting lines $L_2$ and $L_3$ that contain $z$ and support $\mathcal{C}$ on opposite sides, such that $L_1$, $L_2$ and $L_3$ have the centroid property. Furthermore, these are the only intersection points on $L$ for which the centroid property holds. Hence, in this case there are an infinite number of supporting line pairs $L_2$ and $L_3$ which, together with $L_1$, satisfy the centroid property, and all these pairs support $\mathcal{C}$ at $u_j$ and $u_k$, respectively.

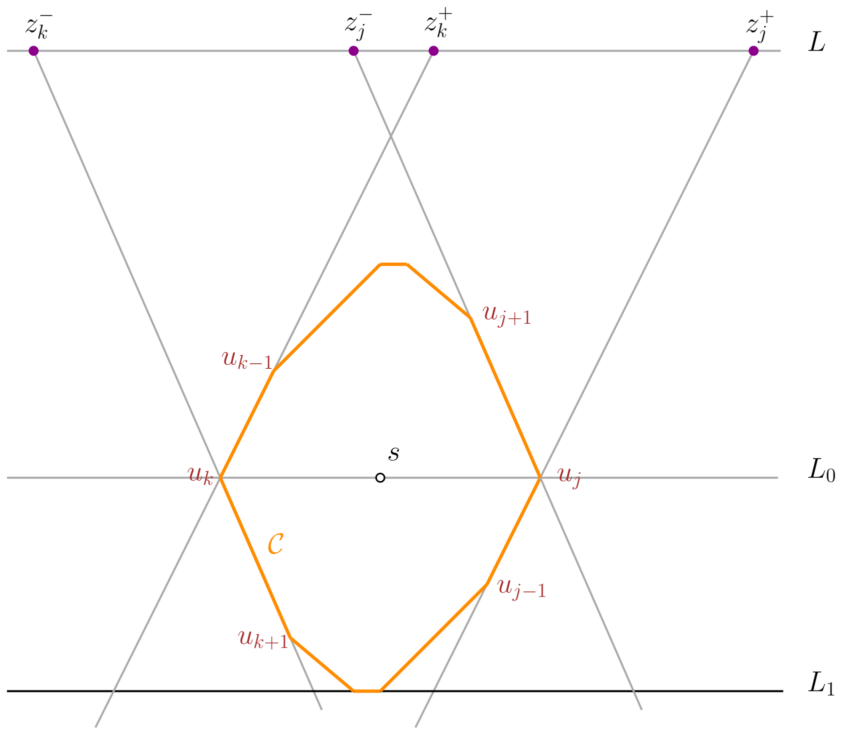


Figure 2.8: An example of a polygonal unit circle $\mathcal{C}$ with supporting line $L_1$ for which $z_j^- < z_k^+$. Here, $L_1$ and any pair of supporting lines through $u_j$ and $u_k$ that intersect in the interval $[z_j^-, z_k^+]$ have the centroid property.

2. $z_j^- = z_k^+$: Here the point $z = z_j^- = z_k^+$ defines a pair of supporting lines $L_2$ and $L_3$ that contain $z$ and support $\mathcal{C}$ on opposite sides, such that $L_1$, $L_2$ and $L_3$ have the centroid property. Note that $L_2$ supports $\mathcal{C}$ both at $u_j$ and $u_{j+1}$ (is a tangent), and similarly $L_3$ supports $\mathcal{C}$ both at $u_k$ and $u_{k-1}$. In this case this is the only pair of supporting lines that, jointly with $L_1$, have the centroid property.

3. $z_j^- > z_k^+$: In this case there is clearly no pair of supporting lines that (jointly with $L_1$) have the centroid property and support $\mathcal{C}$ at either $u_j$ or $u_k$. It follows that the method of proof of statement (a) (Lemma 1.31) applies to this case; hence, exactly one pair of supporting lines, jointly with $L_1$, has the centroid property. The intersection point for this pair of supporting lines lies strictly between $z_k^+$ and $z_j^-$ on $L$. ▮

**Theorem 2.9** *There are at most $4\sigma$ coloured direction sets, where $\sigma$ is the number of legal orientations defined by $\mathcal{C}$. Hence, there are at most $2\sigma$ pairs of complementary coloured direction sets.*[29]

**Proof.** Supppose we fix a pair of adjacent red directions, which correspond to two unit vectors $\mathbf{u}_i$ and $\mathbf{u}_{i+1}$ of $\mathcal{C}$, and let $L_1$ be the line supporting $\mathcal{C}$ at the endpoints of $\mathbf{u}_i$ and $\mathbf{u}_{i+1}$. We will show, using Lemma 1.31 and Lemma 2.8, that there exist either *one* or *two* direction sets for this fixed pair of red directions. For the case considered in Lemma 1.31, where $L_0$ intersects $\mathcal{C}$ at differentiable points, there is exactly *one* direction set which is given by the unit vectors whose endpoints are supported by the unique supporting lines $L_1$, $L_2$ and $L_3$ having the centroid property. If, on the other hand, $L_0$ intersects a vertex of $\mathcal{C}$, then we have the situation considered in Lemma 2.8(b). For the first of the three subcases in the proof of Lemma 2.8 (Figure 2.8) there are exactly two direction sets — one corresponding to a pair of green directions $\mathbf{u}_j$ and $\mathbf{u}_{j+1}$, and a single blue direction $\mathbf{u}_k$; the other corresponding to a single green direction $\mathbf{u}_j$ and a pair of blue directions $\mathbf{u}_{k-1}$ and $\mathbf{u}_k$. The other positions of $L_2$ and $L_3$ do not give direction sets as in each case the corresponding set of directions is not maximal. For the remaining subcases in the proof of Lemma 2.8 there is exactly one direction set. ▮

Note that for $\lambda$-geometry a stronger version of Theorem 2.9 (in which '$4\sigma$' can be replaced by '$2\lambda$') follows almost immediately from the upper and lower bounds on meeting angles (Theorem 2.5). Consider two adjacent directions and a Steiner configuration $\{s, x_1, x_2, x_3\}$ with an edge $sx_1$ that uses these two directions. If $\lambda$ is not a multiple of 3, then, since there are only two feasible meeting angles, both edges $sx_2$ and $sx_3$ must be straight. Hence, for $\lambda$ not a multiple of 3, a direction set contains 4 directions only — one direction set for each pair of adjacent directions. For $\lambda$ a multiple of 3, both edges $sx_2$ and $sx_3$ can be bent while fulfilling the upper and lower bounds on the meeting angles — hence, a direction set contains 6 directions if $\lambda$ is a multiple of 3. Again, only one

| $\lambda$ | Red directions | Green direction(s) | Blue direction(s) |
|---|---|---|---|
| $3m$ | $0,\ \omega$ | $2m\omega,\ (2m+1)\omega$ | $4m\omega,\ (4m+1)\omega$ |
| $3m+1$ | $0,\ \omega$ | $(2m+1)\omega$ | $(4m+2)\omega$ |
| $3m+2$ | $0,\ \omega$ | $(2m+2)\omega$ | $(4m+3)\omega$ |

Table 2.1: Feasible directions in a direction set (up to rotation by a multiple of $\omega$).

direction set is possible if we fix a pair of adjacent directions. Since there are $2\lambda$ pairs of adjacent directions, the theorem follows. Furthermore, these $2\lambda$ direction sets can trivially be constructed in $O(\lambda)$ time. A precise characterisation of the direction sets for each $\lambda$ is given in Table 2.1.

### A quadratic-time algorithm for constructing all direction sets

In this section we give an $O(\sigma^2)$-time algorithm to determine all coloured direction sets for a given centrally symmetric polygonal unit circle $\mathcal{C}$ with $2\sigma$ vertices. Up to symmetry there are $\sigma$ choices of adjacent red directions. By the arguments of Theorem 2.9, there is either one direction set or two direction sets for each fixed pair of red directions. The algorithm iterates over all choices of adjacent red directions, constructing the possible corresponding green and blue directions for each choice in linear time.

Let $\mathbf{u}_i$ and $\mathbf{u}_{i+1}$ be a pair of red directions (we identify directions with the unit vectors of $\mathcal{C}$). In order to determine the direction set(s) for this pair of red directions, we employ the construction used in the proof of Lemma 1.31.

Let $L_1$ be the tangent supporting $\mathcal{C}$ at $u_i$ and $u_{i+1}$, and define $L_0$ and $L$ as in the paragraph immediately preceding Lemma 2.8. In counter-clockwise order around $\mathcal{C}$, starting at $u_{i+1}$, let $u_j$ be the first vertex that is on or above $L_0$. Define $L_2(z)$ as the tangent supporting $\mathcal{C}$ at $u_{j-1}$ and $u_j$ (Figure 2.9), where $z$ denotes the intersection of $L_2(z)$ with $L$. Let $L_3(z)$ be the other supporting line of $\mathcal{C}$ that intersects $z$ (Figure 2.9). Line $L_3(z)$ either supports $\mathcal{C}$ at a single vertex $u_k$, or at two adjacent vertices $u_{k-1}$ and $u_k$. Clearly, given $L_1$, the supporting lines $L_2(z)$ and $L_3(z)$ can be determined in $O(\sigma)$ time.

Now we simulate a continuous movement of the point $z$ to the left along $L$. Let $w_2(z)$ and $w_3(z)$ be the intersections of $L_2(z)$ and $L_3(z)$, respectively, with $L_0$ (Figure 2.9). Note that initially we have $|sw_2(z)| < |sw_3(z)|$, where $s$ is the centre of $\mathcal{C}$. The movement of $z$ is continued until we have a point $z^*$ such that $|sw_2(z^*)| = |sw_3(z^*)|$, that is, until we fulfil the centroid property; by Lemma 1.31 and Lemma 2.8 such a point must exist. At this stage the direction set(s) for the current choice of red edges can be determined by the properties of these two supporting lines through $z^*$.

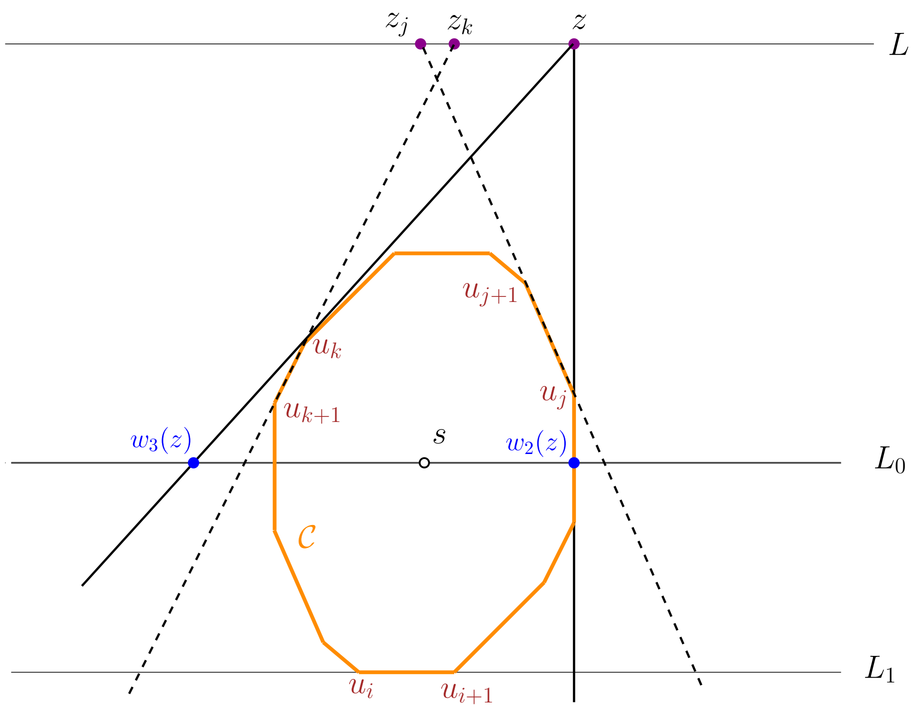


Figure 2.9: Initial step of the algorithm to determine direction sets for a fixed pair of red directions. The diagram shows the initial position of $z$ and the computation of $z_j$ and $z_k$.

A detailed outline of this procedure is given in Algorithm 2.1. The algorithm outputs between $\sigma$ and $2\sigma$ direction sets, each of which is a set of $4$ to $6$ directions (unit vectors of $\mathcal{C}$). The output also includes the correct colour for each direction.

We conclude with a brief discussion of the direction set generation phase of the algorithm. Each direction set contains exactly two red directions and either one or two green and one or two blue directions. The direction sets are determined by whether the supporting lines each touch an edge or only a vertex of $\mathcal{C}$, once the centroid property is fulfilled. There are essentially five possible subcases, depending on the location of $z$ after the completion of the 'while' loop:

**Case 1:** $|sw_2(z)| > |sw_3(z)|$
The point $z^*$ where $|sw_2(z^*)| = |sw_3(z^*)|$ must lie strictly between the last and second last locations of $z$ on $L$. Hence, there is a *unique* direction set with a single green direction and a single blue direction (4 directions in total).

**Case 2:** $|sw_2(z)| = |sw_3(z)|$

(i) $z_j = z$ and $z_k \neq z$: Here we have a *unique* direction set with a pair of green directions and a single blue direction (5 directions in total).

(ii) $z_j \neq z$ and $z_k = z$:

(a) If either $u_j$ or $u_{k+1}$ are not on the line $L_0$ through $s$, then there is a *unique* direction set with a single green direction and two blue directions (5 directions in total).

(b) If, on the other hand, both $u_j$ and $u_{k+1}$ are on the line $L_0$ through $s$, then there are *two* direction sets: the one just described in (i), and one with a pair of green directions and a single blue direction (5 directions in total). Note that in this case we have $u_j = w_2(z)$ and $u_{k+1} = w_3(z)$.

(iii) $z_j = z_k = z$: Here we have a *unique* direction set with two green and two blue directions (6 directions in total).

The algorithm clearly takes $O(\sigma^2)$ time, since the 'while' loop can be executed in constant time, and is iterated at most $2\sigma$ times: in each iteration $j$ and/or $k$ is increased, and there are at most $\sigma$ possibilities for each iterator. Finally, the outer 'for' loop is iterated exactly $\sigma$ times. Hence, we have the following theorem.

**Theorem 2.10** *There are between $\sigma$ and $2\sigma$ direction sets for any given centrally symmetric polygonal unit circle $\mathcal{C}$ with $2\sigma$ vertices, and they can be determined in $O(\sigma^2)$ time.*

Theorem 2.10 can in fact be improved to linear time. The strategy behind the linear-time algorithm is as follows. First we compute a set of supporting lines $L_1$, $L_2$ and $L_3$

**Algorithm 2.1:** Direction sets algorithm

```
Input: A centrally symmetric polygonal unit circle $\mathcal{C}$ with $2\sigma$ ordered vertices
       $u_0, \dots, u_{2\sigma-1}$.
Output: Between $\sigma$ and $2\sigma$ coloured direction sets $D$.

// Iterate over all choices of red directions
for $i = 0$ to $\sigma - 1$ do
    Construct lines $L_1 (= \overline{u_i u_{i+1}})$, $L_0$ and $L$
    Find $j$ such that $u_j$ is the first vertex of $\mathcal{C}$ on or above $L_0$
    Let $z_0$ = $\overline{u_{j-1} u_j} \cap L$, giving the supporting line $L_2(z_0)$
    Find $k$ such that $u_k$ lies on the second supporting line through $z_0$, $L_3(z_0)$
    Let $z_j$ = $\overline{u_j u_{j+1}} \cap L$, and let $z_k$ = $\overline{u_k u_{k+1}} \cap L$
    Let $z$ be the point $z_j$ or $z_k$ closest to $z_0$ (or both if $z_j = z_k$)

    // Simulate movement of $z$
    while $|sw_2(z)| < |sw_3(z)|$ do
        if $z_j = z$ then Set $j = j + 1$
        if $z_k = z$ then Set $k = k + 1$
        Let $z_j$ = $\overline{u_j u_{j+1}} \cap L$, and let $z_k$ = $\overline{u_k u_{k+1}} \cap L$
        Let $z$ be the point $z_j$ or $z_k$ closest to $z_0$ (or both if $z_j = z_k$)

    // Generate direction set(s) for the given red direction
    if $|sw_2(z)| > |sw_3(z)|$ then
        Output $D = (\mathbf{u}_i, \mathbf{u}_{i+1}, \mathbf{u}_j, \mathbf{u}_k)$ as a direction set
    else if $|sw_2(z)| = |sw_3(z)|$ then
        if $z_j = z$ then
            if $z_k \neq z$ then
                Output $D = (\mathbf{u}_i, \mathbf{u}_{i+1}, \mathbf{u}_j, \mathbf{u}_{j+1}, \mathbf{u}_k)$ as a direction set
            else if $z_k = z$ then
                Output $D = (\mathbf{u}_i, \mathbf{u}_{i+1}, \mathbf{u}_j, \mathbf{u}_{J+1}, \mathbf{u}_k, \mathbf{u}_{k+1})$ as a direction set
            else if $z_j \neq z$ and $z_k = z$ then
                Output $D = (\mathbf{u}_i, \mathbf{u}_{i+1}, \mathbf{u}_j, \mathbf{u}_k, \mathbf{u}_{k+1})$ as a direction set
            if $u_j$ and $u_{k+1}$ both lie on $L_0$ then
                Output $D = (\mathbf{u}_i, \mathbf{u}_{i+1}, \mathbf{u}_j, \mathbf{u}_{j+1}, \mathbf{u}_k)$ as a direction set
```

that fulfil the centroid property for a fixed pair of red directions $\mathbf{d}_1 = \mathbf{u}_i$ and $\mathbf{d}_2 = \mathbf{u}_{i+1}$ using the $O(\sigma)$-time algorithm above. Intuitively, we then *rotate* all three supporting lines in counter-clockwise order around $\mathcal{C}$ *while maintaining the centroid property*, locating all positions of the supporting lines that correspond to direction sets. The proof that this can be done efficiently is somewhat technical and will not be included here; details can be found in [74].

The key lemma is to show that only counter-clockwise rotations of the supporting lines are required. Not all supporting lines necessarily rotate in every step, but none of the supporting lines need to rotate *clockwise* in order to maintain the centroid property. Furthermore, in each step of the algorithm — which takes constant time — at least one of the supporting lines will change its rotation point to the successor on $\mathcal{C}$ of the previous rotation point; such a supporting line will after the rotation step be a *tangent* supporting both the previous and new rotation points. Thus, the running time of the algorithm is clearly $O(\sigma)$, since each supporting line has exactly $2\sigma$ possible rotation points.

Note that for most applications the size of $\sigma$ is likely to be relatively small, so optimising the speed of generation of the direction sets is only of marginal importance.

## 2.3 Local properties of full components

The fixed orientation metric is not strictly convex, so in general there are infinitely many minimum Steiner trees for a given set of terminals. In this section we investigate properties of fulsome full Steiner trees, each of which can be thought of as being a component of some larger Steiner tree. (Recall that a Steiner tree is full and fulsome if there is no Steiner tree on the same set of terminals with two or more full components.) The overall aim of this section is to develop a relatively simple canonical form for such Steiner trees that facilitates the design of an efficient algorithm for constructing a Steiner tree for a given topology, given in Section 2.4.

Let $T$ be a full and fulsome fixed orientation Steiner tree. The main results in this section are as follows:

1. We show that $T$ uses a single direction set (Theorem 2.11).
2. We show that any degree $4$ Steiner point in $T$ is a cross (Theorem 2.12).

Next we define *zero-shifts,* which are essentially a way of exchanging primary and secondary material in a pair of edges of $T$ without increasing the length of the tree. A zero-shift is *complete* if it uses all of the exclusively primary or all of the exclusively secondary material from one of the pair of edges. By first examining the properties of fundamental (i.e., indecomposable) zero-shifts we get the following results:

3. Given any two edges $e_1$ and $e_2$ in $T$, where $e_1$ has an exclusively primary segment and $e_2$ has an exclusively secondary segment, we show that there exists a complete zero-shift for $e_1$ and $e_2$ (Theorem 2.19).

4. Using the above, we show that there exists a Steiner tree with the same terminals and topology as $T$ that has at most one bent edge (Corollary 2.20).

The above two results are the key to developing a useful canonical form for $T$ (Theorem 2.23).

We start this process by extending the idea of a direction set for a Steiner point to that of a direction set for a full component.

## 2.3.1 Direction sets

Let $T$ be a full and fulsome minimum fixed orientation Steiner tree where all Steiner points have degree $3$. We begin by describing a (not necessarily unique) method of colouring the edges of $T$.

Pick any Steiner point $s$ and any feasible coloured direction set $\mathcal{D}$ for $s$. The direction set $\mathcal{D}$ defines a colour for each of the edges incident with $s$, and these appear as red, green and blue in counter-clockwise order around $s$. Now pick any Steiner point neighbour $s'$ of $s$. Again assume that the colours appear in counter-clockwise order as red, green and blue around $s'$; thus, the single coloured edge incident with $s'$ uniquely defines the colours of the other two edges. Repeat this procedure until all edges of $T$ have been coloured.

We also assign a parity to the vertices of $T$ depending on whether the path in $T$ from $s$ to that vertex contains an odd or even number of edges. Since $T$ is a tree, this assignment of parity is well defined. When, in the following, we say that there exists a single direction set $\mathcal{D}$ that is used by all Steiner points of $T$, the interpretation should be that the direction set $\mathcal{D}$ is used at even vertices while the complementary direction set of $\mathcal{D}$ is used at odd vertices. This is illustrated in Figure 2.10.

The following key theorem is similar in spirit to Theorem 1.33 for metrics defined by differentiable unit circles, which says that edges of a full Steiner tree use at most three different orientations. Here we show that for a polygonal unit circle the Steiner points in a full and fulsome Steiner tree all use a single direction set.

**Theorem 2.11** *Given a full and fulsome minimum fixed orientation Steiner tree $T$, there exists a single direction set that is used by every Steiner point in $T$.*

**Proof.** We prove the theorem by showing that, given any two adjacent Steiner points $s_1$ and $s_2$ in $T$, there exists a single direction set that is used by $s_1$ and $s_2$. Furthermore,

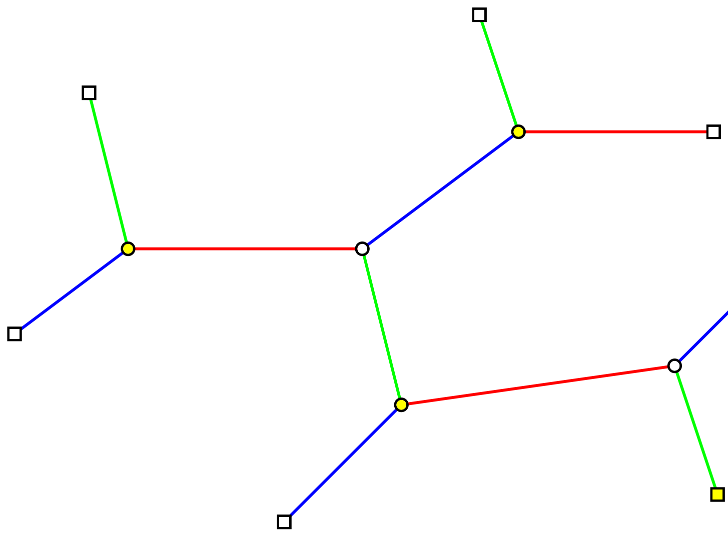


Figure 2.10: Example of a full Steiner tree showing an assignment of colours to edges. The different colours of the nodes (yellow or white) correspond to the assignment of parity.

we show that if the edges incident with $s_1$ and the edges incident with $s_2$ do not share exactly the same set of directions, then there exists a small finite perturbation of $s_1$ and $s_2$ such that the resulting tree is still a full minimum fixed orientation Steiner tree and the directions used by the edges incident with each of $s_1$ and $s_2$ exactly coincide. This means that the direction set at any Steiner point can be propagated throughout the tree, since the Steiner points of a full Steiner tree induce a tree. The theorem then immediately follows.

Let $s_1$ and $s_2$ be adjacent Steiner points in $T$. Let $v_1$ and $v_2$ be the nodes or corner points adjacent to $s_1$ on the other two incident edges, travelling counter-clockwise from $s_1s_2$, i.e., line segments $s_1v_1$ and $s_1v_2$ each use a single legal orientation. Similarly, let $v_3$ and $v_4$ be the nodes or corner points adjacent to $s_2$ on the other two incident edges, travelling counter-clockwise from $s_2s_1$. Let $T_1$ be the resulting full Steiner tree on terminal set $\{v_1, v_2, v_3, v_4\}$ with Steiner points $s_1$ and $s_2$. This is illustrated in Figure 2.11(a). Let $\mathbf{u}_1$ and $\mathbf{u}_2$ be the directions of $\overrightarrow{s_1v_1}$ and $\overrightarrow{s_1v_2}$, respectively, and let $\mathbf{u}_3$ and $\mathbf{u}_4$ be the directions of $\overrightarrow{s_2v_3}$ and $\overrightarrow{s_2v_4}$, respectively.

For simplicity, we assume either that $s_1s_2$ is a straight edge or that it is embedded using two corner points (like the left-hand path in Figure 2.2), so that both ends of the edge use the same direction. Let $\theta_1, \theta_2, \theta_3$ be the three angles around $s_1$, travelling counter-clockwise from $s_1s_2$, and let $\phi_1, \phi_2, \phi_3$ be the three angles around $s_2$, travelling counter-clockwise from $s_2s_1$. Again, this is illustrated in Figure 2.11(a). By the sliding lemma from Chapter 1 (Lemma 1.37), we can choose the embedding of the original full Steiner

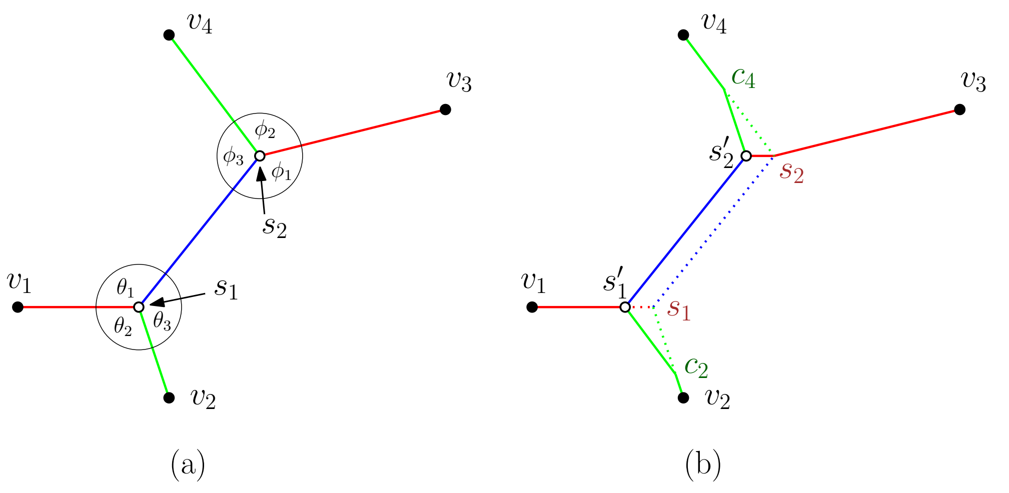


Figure 2.11: Performing a shift on adjacent Steiner points.

tree $T$ such that

$$\theta_1 + \phi_3 > \pi. \tag{2.1}$$

Suppose that

$$\theta_3 > \phi_3. \tag{2.2}$$

Under this assumption, we construct a transformation on $T_1$ that does not increase its length. We do this by defining a *shift* on $s_1$ and $s_2$ (shifts will be discussed in more generality later in this section). This involves moving each of $s_1$ and $s_2$ a distance $\varepsilon$ in direction $\mathbf{u}_1$ to $s'_1$ and $s'_2$, respectively, where $\varepsilon > 0$ is small compared to all edge lengths in $T_1$.

We claim that we can construct a line segment from $s'_1$ in direction $-\mathbf{u}_4$ meeting $s_1v_2$ at a point $c_2$. For such a construction to be possible, for sufficiently small $\varepsilon$, we require that the ray from $s_1$ in direction $\mathbf{u}_2$ must intersect the ray from $s'_1$ in direction $-\mathbf{u}_4$. This occurs if $\phi_3 \geq \pi - \theta_1$ and $\phi_3 < \theta_3$ which follow from inequalities (2.1) and (2.2), respectively. Similarly, we can construct a line segment from $s'_2$ in direction $-\mathbf{u}_2$ meeting $s_2v_4$ at a point $c_4$. Furthermore, $c_4$ does not coincide with $s'_2$ since inequality (2.1) is a strict inequality.

Now, construct a new tree $T_2$ interconnecting $\{v_1, v_2, v_3, v_4\}$ via Steiner points $s'_1$ and $s'_2$, such that $c_2$ is the corner point of the edge $s'_1v_2$, $c_4$ is the corner point of the edge $s'_2v_4$ and $s_2$ is the corner point of the edge $s'_2v_3$. The remaining external edge, $s'_1v_1$, is a straight edge. This is illustrated in Figure 2.11(b).

We next observe that $\|T_2\| = \|T_1\|$. To see this, note that in transforming $T_1$ to $T_2$ the edge $s_1s_2$ and line segments $s_1s_1'$, $s_1c_2$ and $s_2c_4$ have been removed, and the edge $s_1's_2'$ and line segments $s_2s_2'$, $s_2'c_4$ and $s_1'c_2$ have been added. But $\|s_1s_2\| = \|s_1's_2'\|$ (since $s_1$ and $s_2$ undergo identical translations), $\|s_1s_1'\| = \|s_2s_2'\|$ (by construction) and $\|s_1c_2\| = \|s_2'c_4\|$, $\|s_2c_4\| = \|s_1'c_2\|$ (since triangles $\Delta s_1s_1'c_2$ and $\Delta s_2's_2c_4$ are congruent). Hence, $\|T_2\| = \|T_1\|$. In other words, $T_2$ is also a full Steiner tree on the terminal set of $T_1$.

We now show that there is a single direction set that is used by $s_1$ and $s_2$ in the original full Steiner tree, and that by performing a pair of shifts as described above we can construct a new Steiner tree such that the directions used by the edges incident with $s_1$ exactly coincide with the directions used by the edges incident with $s_2$. We do this by showing that, for each of the three colour labels for edges, the direction sets for $s_1$ and $s_2$ coincide on that colour and we can ensure that all directions are used at each Steiner point. Note that this is trivially true for the colour of the edge $s_1s_2$. Now, observe that the edges (or half-edges) $s_1v_2$ and $s_2v_4$ are labelled with the same colour (say, blue). If both these line segments use the same direction for every embedding, then the direction sets for $s_1$ and $s_2$ coincide on the blue colour. If there exists an embedding such that $s_1v_2$ and $s_2v_4$ have different directions, then either $\theta_3 > \phi_3$ or $\phi_3 > \theta_3$. In either case we can perform the local transformation above (swapping the roles of $s_1$ and $s_2$ if necessary), resulting in a new Steiner tree that uses both blue directions at $s_1'$ and $s_2'$. In order for this new tree to be minimal (under any embedding of the original tree) it again follows that the direction sets for $s_1$ and $s_2$ coincide on the blue label, and that both directions are used by both edges after applying the transformation. The same argument applies to the remaining colour label, concluding the proof. ∎

A corollary of this theorem is that the edges of a full and fulsome minimum fixed orientation Steiner tree use at most six legal orientations. Later in this section we show that four legal orientations actually suffice.

### 2.3.2 Degree $4$ Steiner points

Over the next few sections, we will develop an understanding of the geometric structure of minimum fixed orientation Steiner trees by studying the effects of length-preserving perturbations on the tree, similar to that used in the proof of Theorem 2.11. If a full and fulsome Steiner tree $T$ has more than four terminals, then by Theorem 1.39 we can initally assume that all Steiner points have degree $3$; hence, there exists a direction set and colouring for the edges of $T$. However, during the process of moving the Steiner points some edges may degenerate to length zero, creating degree $4$ Steiner points. The following theorem gives strong restrictions on when such a Steiner point can occur as part of a fulsome Steiner tree. This theorem builds on some of the properties of degree $4$ Steiner points developed in Section 1.6 of Chapter 1.

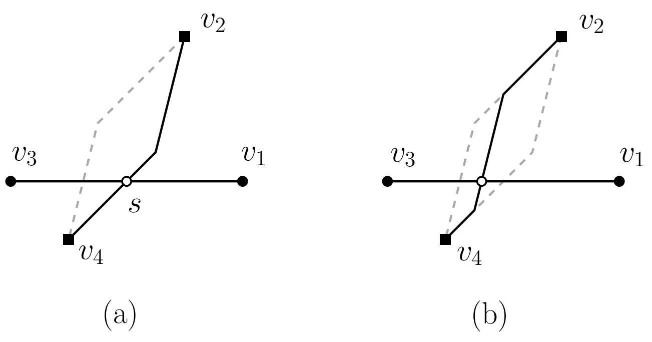


Figure 2.12: A degree $4$ Steiner point $s$ that is not a cross. By perturbing $s$ if necessary, the second pair of opposite edges can be embedded so as to be locally collinear at the Steiner point in two different ways, as shown in (a) and (b).

Recall that a *cross* is defined to be a degree $4$ Steiner point where both opposite pairs of edges are collinear. Hence, in the context of fixed orientation trees, all four incident edges must also be straight edges.

**Theorem 2.12** *In a fulsome Steiner tree, a degree $4$ Steiner point must be a cross,* unless *it is adjacent to terminals only.*

**Proof.** Consider a degree $4$ Steiner point $s$ with neighbours $v_1$, $v_2$, $v_3$ and $v_4$ which does not form a cross. By Lemma 1.35, one pair of opposite edges (say $(s, v_1)$ and $(s, v_3)$) are collinear and straight, but by the assumption the second pair of opposite edges, $(s, v_2)$ and $(s, v_4)$, are not collinear and possibly not straight (as in Figure 2.12(a)).

By Lemma 1.36 and the discussion that follows it, the Steiner point $s$ can be split into two adjacent degree $3$ Steiner points (as illustrated in Figure 2.13(a) and (b)). Hence, by Lemma 1.38 vertices $v_2$ and $v_4$ are terminals. In such a splitting, the edges incident with $v_2$ and $v_4$ both have the same colour (say red), and hence these edges use at most two adjacent orientations.

Before we prove that $v_1$ and $v_3$ also must be terminals, we make a few observations. Two embedded fixed orientation edges incident with a single point are said to be *locally collinear* if the straight line segments of the edges immediately incident with the point are collinear. By moving $s$ along the line segment $v_1v_3$, we can find embeddings that make the second pair of opposite edges locally collinear in two different ways as shown in Figure 2.12(a) and Figure 2.12(b). As a consequence, the vertex $v_1$ cannot be a degree $4$ Steiner point, since then we could construct a pair of locally collinear edges around $s$ and $v_1$, respectively, but the collinear edges of $s$ and $v_1$ could be made non-parallel — hence making a length-decreasing shift of the edge $(s, v_1)$ possible.

Thus, if $v_1$ is a Steiner point, then it will be a degree $3$ Steiner point. Let $x_1$ and $x_2$ be the two neighbours of $v_1$ (other than $s$), where $x_1$ is on the same side of $\overline{v_1v_3}$ as $v_2$

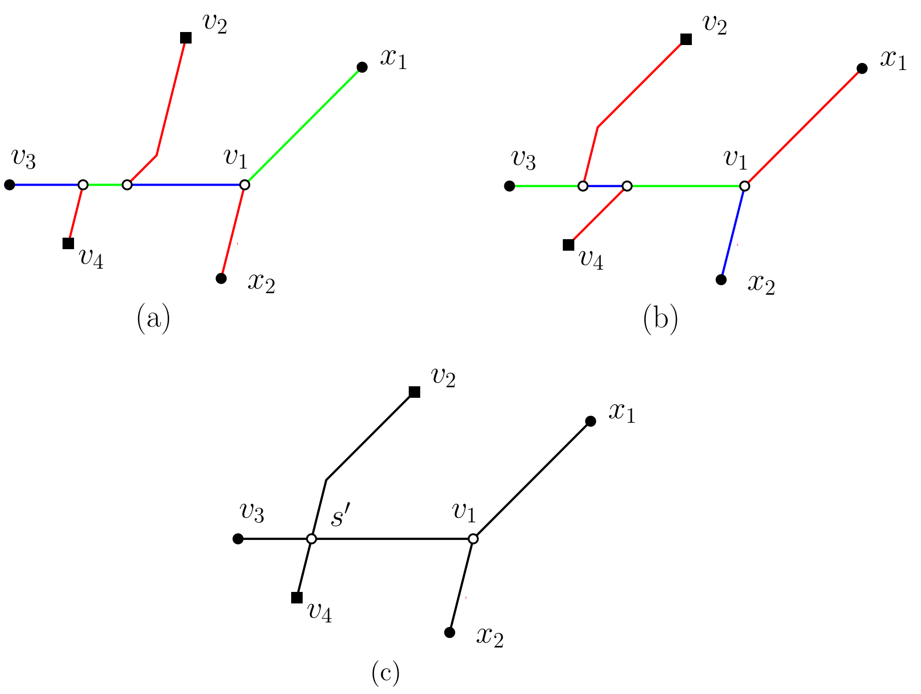


Figure 2.13: Diagrams (a) and (b) show two topologically distinct splittings of $s$, and the resulting colours of the nearby edges. Note that both $(v_1, x_1)$ and $(v_1, x_2)$ must use red orientations. Diagram (c) illustrates that there is now a locally minimal location of $s$ that leads to a contradiction to fulsomeness by the sliding lemma.

and $x_2$ is on the same side of $\overline{v_1v_3}$ as $v_4$. By splitting $s$ into degree $3$ Steiner points in the two topologically different ways shown in Figure 2.13(a) and 2.13(b), and applying Theorem 2.11, it follows that the two edges $(v_1, x_1)$ and $(v_1, x_2)$ must use the same pair of adjacent legal orientations as $(s, v_2)$ and $(s, v_4)$. If either one of the edges $(v_1, x_1)$ or $(v_1, x_2)$ is a bent edge, then we can always construct a pair of locally collinear edges at $v_1$. This again leads to a contradiction to length minimality by using the same arguments as in the degree $4$ case.

As a consequence, we are left with the case where the edges $(v_1, x_1)$ and $(v_1, x_2)$ are straight and not collinear, as in the figure. However, it is possible to move $s$ along the line segment $v_1v_3$ to a point $s'$ where either $(s', v_2)$ is straight and parallel to $(v_1, x_1)$ — or $(s', v_4)$ is straight and parallel to $(v_1, x_2)$ (Figure 2.13(c)). In either case, by Lemma 1.37, we arrive at a contradiction to fulsomeness. ∎

### 2.3.3 Zero-shifts

The efficiency of the algorithms that we develop in this chapter for constructing Steiner trees comes from the fact that we can assume that full Steiner trees have particular canonical forms. Our means of establishing these canonical forms is to use the properties of length-preserving perturbations, denoted *zero-shifts*, which we describe in this section.[30]

#### Fundamental zero-shifts

By Theorem 2.11, we can think of a coloured direction set $\mathcal{D} = \{\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_k\}$ as applying to all edges in a full and fulsome Steiner tree $T$. Recall that the elements of $\mathcal{D}$, treated as vectors rooted at the origin, appear in counter-clockwise order, beginning with $\mathbf{d}_1$, which corresponds to the exclusively primary red direction, and $\mathbf{d}_2$, which corresponds to the exclusively secondary red direction.

**Definitions [Zero-shifts]**: We define a *zero-shift* to be a perturbation of one or more Steiner points in $T$ that does not increase the length of $T$. Such a perturbation $\zeta$ is called a *fundamental zero-shift* if it cannot be decomposed into two zero-shifts, each of which acts on a subset of the Steiner points acted on by $\zeta$, and at least one of which acts on a proper subset of those Steiner points.

Note that if a zero-shift exists in a tree $T$, then we can continue to apply this shift, without changing the length of $T$, until the length of some line segment of $T$ goes to zero (for example, until a Steiner point coincides with a corner point or with another Steiner point). This means that we can treat zero-shifts as discrete repositionings of Steiner points,

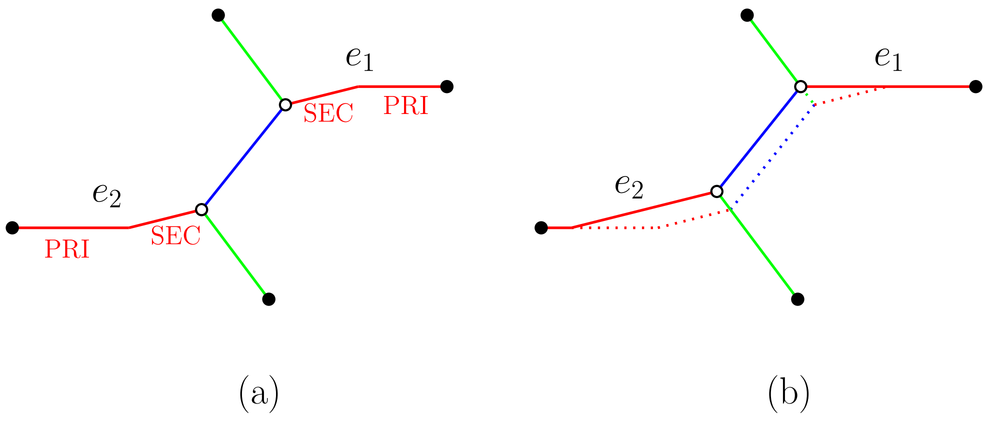


Figure 2.14: An example of a complete zero-shift between edges $e_1$ and $e_2$. Diagram (a) shows the initial state, where exclusively primary and exclusively secondary segments are labelled 'PRI' and 'SEC', respectively. Diagram (b) shows the consequence of a complete zero-shift, resulting in edge $e_1$ being exclusively primary. Note that directions of the straight edges in (a) are preserved by the zero-shift.

rather than simply perturbations. The importance of such a zero-shift is that it can effectively move exclusively primary and exclusively secondary material from one edge to another without increasing the length of the tree $T$. This idea motivates the following definition.

**Definition [Complete zero-shifts]**: Let $e_1$ and $e_2$ be two edges in $T$ such that $e_1$ has an exclusively secondary segment and $e_2$ has an exclusively primary segment. Suppose there exists a zero-shift $\zeta$ on the Steiner points on the path between $e_1$ and $e_2$. Then $\zeta$ is called a *complete zero-shift* for $e_1$ and $e_2$ if after applying $\zeta$ either $e_1$ is exclusively primary or $e_2$ is exclusively secondary, and $\zeta$ preserves the direction of all straight edges in $T$ except (possibly) $e_1$ and $e_2$.

Figure 2.14 shows an example of a complete zero-shift.

We will show that complete zero-shifts exist for any suitable pairs of edges in $T$. We begin by briefly categorising those fundamental zero-shifts that act on either one Steiner point or two adjacent Steiner points in $T$. These are referred to, respectively, as 1-point and 2-point fundamental zero-shifts. Later we will in fact show that these are the only fundamental zero-shifts that can occur in $T$. We first require the following lemma.

**Lemma 2.13** *If a direction set $\mathcal{D}$ has cardinality $4$, then the angle between the green and blue directions is strictly less than $\pi$.*

For the proof of this lemma, see Exercise 2.7.

Let $s$ be a Steiner point in $T$ such that the edges incident with $s$ use all directions in $\mathcal{D}$. Clearly, for any $\mathcal{D}$, there exists a $T$ that contains such a Steiner point; for example, we can construct a suitable Steiner configuration $T$ with this property.

We now consider properties of 1-point fundamental zero-shifts for $s$. We show that the existence of such a perturbation depends on the number of directions in the direction set for $s$.

**Lemma 2.14** *Let $s$ be a Steiner point in $T$ such that the edges incident with $s$ use all directions in some given direction set $\mathcal{D}$. Let $k$ be the cardinality of $\mathcal{D}$.*

*(a) If $k = 4$ then there is no 1-point zero-shift of $s$.*

*(b) If $k = 5$, where $\mathbf{d}_5$ is the only blue direction, then there exist 1-point fundamental zero-shifts for $s$, perturbing $s$ in the directions of $\mathbf{d}_5$ and $-\mathbf{d}_5$ only.*

*(c) If $k = 6$ then there exist 1-point fundamental zero-shifts for $s$, perturbing $s$ in any direction.*

**Proof.** For statement (a), we observe, by Lemma 2.13, that the angle $\theta$ between the green and blue directions satisfies $0 < \theta < \pi$. It follows that the position of $s$ is uniquely determined by the positions of the two adjacent vertices incident with the green and blue edges, and thus there can be no 1-point zero-shift of $s$.

For statements (b) and (c) it is straightforward to confirm the existence of suitable 1-point zero-shifts. For the case $k = 5$, a zero-shift of $s$ in the direction of $-\mathbf{d}_5$ to a new Steiner point $s'$ is illustrated in Figure 2.15.

The new point $s'$ uses the same direction set as $s$ and hence is still part of a minimum Steiner configuration. In the same way, the shift from $s'$ to $s$ is a zero-shift in the direction of $\mathbf{d}_5$. Clearly, there are no possible zero-shifts in other directions, as that would create a second blue direction. For $k = 6$ we can construct independent zero-shifts of this type in directions $\mathbf{d}_5$ and $\mathbf{d}_6$, and hence in any direction, since those two vectors span $\mathbb{R}^2$. ∎

The existence of the zero-shifts given in Lemma 2.14 (b) and (c) imply that a direction set with cardinality $5$ or $6$ can only exist if the directions and weights satisfy a prescribed relationship; see Exercise 2.8.

The next lemma shows that 1-point fundamental zero-shifts can always be made complete.

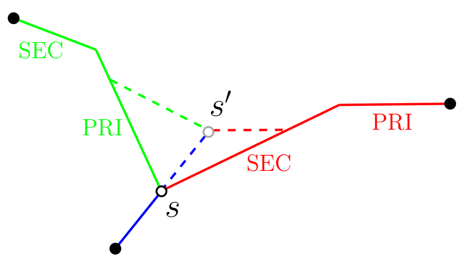


Figure 2.15: A $1$-point fundamental zero-shift for a Steiner point whose direction set contains $5$ directions. Here, exclusively primary and exclusively secondary segments of the original tree are labelled 'PRI' and 'SEC', respectively.

**Lemma 2.15** *Let $e_1$ and $e_2$ be two edges in a full and fulsome minimum fixed orientation Steiner tree $T$ both incident with a degree $3$ Steiner point $s$, such that $e_1$ has an exclusively secondary segment and $e_2$ has an exclusively primary segment. Then there exists a complete $1$-point zero-shift for $e_1$ and $e_2$ in $T$.*

**Proof.** Let $v$ be the neighbouring vertex to $s$ that is not an endpoint of $e_1$ or $e_2$. Since $e_1$ and $e_2$ have different colours, it follows from the conditions of the lemma that the direction set for $s$ has cardinality at least $5$. Hence, by Lemma 2.14, there exists a $1$-point zero-shift that either moves $s$ away from or towards $v$ while decreasing both the secondary segment of $e_1$ and the primary segment of $e_2$. If $s$ moves away from $v$, then the lemma clearly holds. If $s$ moves towards $v$, then the only problem that can occur is that $s$ may meet $v$ before either $e_1$ or $e_2$ is straight. But in that case $T$ is either not fulsome (if $v$ is a terminal) or we have two adjacent bent edges incident with a degree $4$ Steiner point, which is a contradiction to $T$ being a full Steiner tree, by Lemma 1.35. ∎

Next, we consider properties of $2$-point fundamental zero-shifts.

**Lemma 2.16** *Let $s_1$ and $s_2$ be neighbouring Steiner points in a full and fulsome minimum fixed orientation Steiner tree $T$. Assume that $(s_1, s_2)$ is a straight edge which is neither exclusively primary nor exclusively secondary. Let $e_1$ and $e_2$ be distinct edges of $T$ incident with $s_1$ and $s_2$, respectively, such that $e_1$ and $e_2$ have the same colour in the direction set of $T$. Assume that for at least one of $i = 1$ or $i = 2$ the meeting angle between the two edges other than $e_i$ at $s_i$ is not $\pi$. For each $i \in \{1, 2\}$ let $v_i$ be the closest neighbouring node or corner point on $e_i$ to $s_i$. If $\overrightarrow{s_1 v_1}$ and $\overrightarrow{v_2 s_2}$ have different directions, then there exists a $2$-point fundamental zero-shift for $s_1$ and $s_2$.*

**Proof.** This lemma follows immediately from the proof of Theorem 2.11, using the construction illustrated in Figure 2.11. Note that the condition on $(s_1, s_2)$ means that the direction set of $T$ contains either $4$ or $5$ directions. It follows that there are no 1-point zero-shifts that move $s_1$ or $s_2$ in the same direction as the constructed 2-point zero-shift, and hence that the 2-point zero-shift is fundamental. ∎

There are two remaining 2-point zero-shifts not covered by Lemma 2.16. The first is where $(s_1, s_2)$ contains an exclusively primary or exclusively secondary segment; in this case there is a 1-point fundamental zero-shift at either $s_1$ or $s_2$, and hence any 2-point zero-shift on these Steiner points is not fundamental. The second is where $T$ contains meeting angles of $\pi$ at both $s_1$ and $s_2$, in each case between the two incident edges other than $e_1$ or $e_2$. This implies that all edges incident with $s_1$ and $s_2$ other than $e_1$ and $e_2$ are straight and collinear. In this case, the resulting 2-point zero-shift is again not fundamental, but can be decomposed into two independent 1-point zero-shifts at each Steiner point.

## General zero-shifts

We next show that 2-point fundamental zero-shifts, and indeed general zero-shifts, can be made complete. A potential difficulty in doing this comes from the possible formation of degree $4$ Steiner points, during a zero-shift. By Theorem 2.12, in a fulsome Steiner tree $T$ a degree $4$ Steiner point must be a cross, unless it is adjacent to terminals only.

We begin with two useful corollaries to this theorem.

**Corollary 2.17** *Let $e_1$ and $e_2$ be two distinct edges in $T$ incident with neighbouring Steiner points, $s_1$ and $s_2$, respectively. Suppose that $e_1$ and $e_2$ have the same colour in the direction set of $T$, and that $e_1$ has an exclusively secondary segment and $e_2$ has an exclusively primary segment. Then there exists a complete $2$-point zero-shift for $e_1$ and $e_2$.*

**Proof.** First, suppose that the edge $(s_1, s_2)$ is straight. Then we can apply a 2-point zero-shift of the sort illustrated in Figure 2.11(b) (where $(s_1, v_2) = e_1$ and $(s_2, v_4) = e_2$). This zero-shift acts to decrease the length of one of the edges $e_0$ incident with $s_1$ and $s_2$ but not on the path from $e_1$ to $e_2$ (where $(s_1, v_1) = e_0$ in the figure). It immediately follows that the zero-shift can be continued until it is complete, unless the length of $e_0$ decreases to $0$ before either $e_1$ or $e_2$ is straight. However, in such a case the resulting tree contains a degree $4$ Steiner point with an incident bent edge, and a second Steiner point, giving a contradiction to fulsomeness by Theorem 2.12.

If on the other hand, $(s_1, s_2)$ is a bent edge, then we can embed it with two corner points (as in the path in Figure 2.2), and then apply the same zero-shift and argument as above. ∎

Note that Corollary 2.17 implies that any $2$-point fundamental zero-shift can be made complete.

**Corollary 2.18** *Suppose $T$ has four terminals and a single (degree $4$) Steiner point $s$. Let $e_1$ and $e_2$ be two edges in $T$ such that $e_1$ has an exclusively secondary segment and $e_2$ has an exclusively primary segment. Then there exists a complete $1$-point zero-shift for $e_1$ and $e_2$.*

The proof of this corollary is straightforward (Exercise 2.9).

The following theorem gives conditions for the existence of complete general zero-shifts. This theorem is the key tool for constructing canonical forms in the next section.

**Theorem 2.19** ***[Complete Zero-Shift Theorem]*** *Let $e_1$ and $e_2$ be two edges in a full and fulsome Steiner tree $T$ such that $e_1$ has an exclusively secondary segment and $e_2$ has an exclusively primary segment. Then there exists a zero-shift acting on the Steiner points on the path from $e_1$ to $e_2$ in $T$ that is complete for $e_1$ and $e_2$.*

**Proof.** Let $s_1$ and $s_2$ be, respectively, the first and last Steiner points on the path from $e_1$ to $e_2$ in $T$. We may assume $s_1$ and $s_2$ are distinct, since otherwise the theorem is immediately true by Lemma 2.15 and Corollary 2.18. We may also assume that $s_1$ and $s_2$ each have degree $3$ in $T$, by Theorem 1.39. For each $i \in \{1, 2\}$ let $\theta_i$ be the meeting angle at $s_i$ between the two incident edges other than $e_i$.

**Case 1**. For $i = 1$ or $i = 2$ assume that the two edges incident with $s_i$ are straight edges and that $\theta_i = \pi$. Since $e_i$ has an exclusively primary or secondary segment, there is a 1-point zero-shift moving $s_i$ along the line through those other two edges (as implied by Lemma 1.36). If $T$ spans at least five terminals and the moving Steiner point $s_i$ meets another Steiner point before the edge $e_i$ becomes straight, then this is a contradiction to fulsomeness by Theorem 2.12. If, on the other hand, $T$ spans four terminals and a Steiner point $s_i$ meets another Steiner point before the zero-shift is complete, then in fact we must have $s_1 = s_2$, and the zero-shift can be made complete by Corollary 2.18. Hence, it is always possible either to make $e_1$ exclusively primary or to make $e_2$ exclusively secondary.

**Case 2**. If Case 1 does not apply, then we argue by induction on the number of Steiner points in the path between $e_1$ and $e_2$. As a first step, we establish some base cases for the induction, where there are at most two Steiner points in the path from $e_1$ to $e_2$, in other words, for 1-point and 2-point zero-shifts.

If $s_1 = s_2$, or if $e_1$ and $e_2$ have the same colour and are incident with neighbouring Steiner points, then the existence of a complete zero-shift follows from Lemma 2.15 and Corollary 2.18, respectively. Hence, the only base case remaining to consider is the one in which $e_1$ and $e_2$ have different colours and $s_1$ and $s_2$ are neighbours. Let $e_0$ be the edge incident with $s_2$ with the same colour as $e_1$. If $e_0$ is exclusively primary, then an appropriate zero-shift can be constructed as follows (see Figure 2.16).

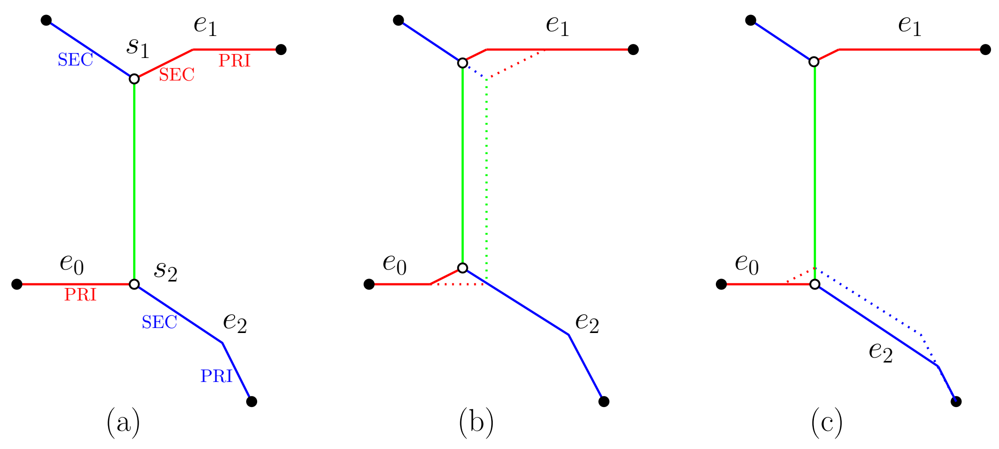


Figure 2.16: An example of a 2-point zero-shift between two edges $e_1$ and $e_2$ with different colours. Diagram (a) shows the initial state, where exclusively primary and exclusively secondary segments are labelled 'PRI' and 'SEC', respectively. In (b) a small 2-point fundamental zero-shift transfers some secondary material from $e_1$ to $e_0$. In (c) a 1-point fundamental zero-shift transfers all of this secondary material from $e_0$ to $e_2$. The result is that the secondary/primary material ratio has been strictly reduced in $e_1$ and increased in $e_2$.

We can apply a small 2-point fundamental zero-shift at $s_1$ and $s_2$, effectively transferring an arbitrarily small exclusively secondary segment to $e_0$. In particular, it strictly reduces the secondary/primary material ratio in $e_1$, and increases it in $e_0$. Now there exists a 1-point fundamental zero-shift between the secondary segment of $e_0$ and the edge $e_2$. Because the secondary segment of $e_0$ can be assumed to be arbitrarily small, it follows that this second shift can reduce the secondary segment of $e_0$ to zero. Hence, together these two shifts form a zero-shift on the path between $e_1$ and $e_2$ that preserves the direction of $e_0$. By Theorem 2.12 and the fulsomeness of $T$ this combined zero-shift can continue to be applied until it is complete for $e_1$ and $e_2$. The only remaining subcase is where $e_0$ has an exclusively secondary segment, in which case we apply a similar argument, but reverse the order of the two fundamental zero-shifts.

We now conclude the argument by applying induction on the number of Steiner points in the path between $e_1$ and $e_2$. The inductive step involves generalising the construction in the previous paragraph, where $e_0$ is now any suitable edge incident to a Steiner point on the path between $e_1$ and $e_2$. An easy but important observation in the inductive step is that we can always find a suitable edge $e_0$ incident with some $s_0$ on the path between $e_1$ and $e_2$ such that the condition of Case 1 ($\theta_0 = \pi$) does not apply at $s_0$. This means that the required increase and decrease in secondary/primary material ratio occurs in the smaller zero-shift by the inductive assumptions and can be continued until the shift is complete. The induction argument now follows. ∎

Note that an immediate corollary of the proof of Theorem 2.19 is that any zero-shift, other than the $1$-point and $2$-point fundamental zero-shifts described earlier, can be decomposed into two zero-shifts, at least one of which acts on a strictly smaller set of Steiner points. Hence, the only fundamental zero-shifts are those described in Lemmas 2.14 and 2.16.

Another important corollary of Theorem 2.19 is the following.

**Corollary 2.20** *Let $N$ be a given set of terminals, and suppose there exists a full and fulsome fixed orientation Steiner tree $T$ for $N$. Then there exists a full fixed orientation Steiner tree for $N$ (with the same topology as $T$) which contains at most one bent edge.*

**Proof.** The corollary is an immediate application of Theorem 2.19. If $T$ contains two bent edges $e_1$ and $e_2$, perform a zero-shift acting on the Steiner points on the path in $T$ between these edges that is complete for $e_1$ and $e_2$. This reduces the number of bent edges in $T$ by at least $1$. The corollary follows by repeating this procedure until there is no remaining pair of bent edges. ∎

A further property of zero-shifts that will prove useful in the later sections is that in almost all cases they preserve the total amount of primary material in a fixed orientation Steiner tree $T$. Note that if $T$ contains directions that are both primary and secondary we say that the amount of primary material is *preserved* under a zero-shift if there exists a partitioning of each non-exclusive edge of $T$ into primary and secondary segments, both before and after the zero-shift, so that the total length of 'primary' edges remains unchanged.

**Theorem 2.21** ***[Primary material preserved]*** *Let $T$ be a fulsome minimum fixed orientation Steiner tree such that $T$ is either a $\lambda$-geometry Steiner tree or the direction set for $T$ contains at most $5$ directions. Then the total amount of primary material in $T$ is preserved under zero-shifts.*

**Proof.** Clearly, it suffices to show that the total amount of primary material in the tree is preserved by any fundamental zero-shift. For a $2$-point fundamental zero-shift the result immediately follows from the description of such shifts in Lemma 2.16. For a $1$-point fundamental zero-shift where there are exactly $5$ directions the result follows from the definition of preservation of primary material above, since there exists a non-exclusive edge incident with the Steiner point where the fundamental zero-shift takes place. Finally, if $T$ is a $\lambda$-geometry Steiner tree with $6$ distinct directions in its direction set, then $\lambda$ is a multiple of $3$ (see Table 2.1), and the preservation of primary material can be shown by direct construction (Exercise 2.10). ∎

The only case where Theorem 2.21 may fail is when the direction set has $6$ directions and the tree is not a $\lambda$-geometry Steiner tree. This is a highly contrived situation, which will almost never occur in practice; for example, the weights must satisfy the conditions given in Exercise 2.8(b). In what follows we will ignore this case and assume that we can apply the theorem.

The next corollary gives a condition under which a given edge in a minimum Steiner tree can be identified as being able to be the unique bent edge of the tree (after some suitable zero-shifts). This will prove to be particularly useful in Section 2.5.3 of this chapter.

**Corollary 2.22** *Let $T$ be a full and fulsome minimum fixed orientation Steiner tree on a given set of terminals $N$ such that $T$ is either a $\lambda$-geometry Steiner tree or the direction set for $T$ contains at most $5$ directions. Let $e$ be an edge of $T$, such that $e$ is red if $\lambda \neq 3m$ (or any colour if $\lambda = 3m$). Suppose that the total length of all exclusively primary (or exclusively secondary) components of the edges of $T$ is small compared to $|e|$. Then there exists a full and fulsome minimum Steiner tree for $N$ with the same topology as $T$ such that the only edge of this tree containing an exclusively primary (respectively, secondary) component is the edge corresponding to $e$.*

This is an easy consequence of Theorems 2.19 and 2.21 obtained by performing a series of zero-shifts, each of which moves some exclusively primary (respectively, secondary) material to $e$. If the length of $e$ is sufficient to absorb all such material in the tree, then the corollary follows.

### 2.3.4 Canonical forms

The existence of zero-shifts makes it possible to transform any full and fulsome Steiner tree into another full Steiner tree with the same topology that has some well-defined canonical form. In this section we establish canonical forms for Steiner trees for the fixed orientation metric, based on the form originally proposed in [70] for the uniform orientation metric.

**Definitions [Canonical tree, transition edge]**: Given an ordering of the edges in a full Steiner topology $\mathcal{T}$, a full Steiner tree $T$ for topology $\mathcal{T}$ is said to be *canonical* with respect to that ordering if $T$ contains an edge, which we refer to as a *transition edge*, satisfying the following properties:

- all edges other than the transition edge are straight edges;
- all edges that come before the transition edge under the given ordering are primary;
- all edges that come after the transition edge under the given ordering are secondary.

Note that this definition implies that a canonical tree has at most one bent edge.

The proof of the following theorem follows from the fact that we can use zero-shifts to move primary/secondary material between edges in $T$.

**Theorem 2.23** *Let a set of terminals $N$ and a full Steiner topology $\mathcal{T}$ for that set of terminals be given. Suppose there exists a full and fulsome fixed orientation Steiner tree for $N$ with topology $\mathcal{T}$. Then for any ordering of the edges of $\mathcal{T}$ there exists a full fixed orientation Steiner tree for $N$ (with topology $\mathcal{T}$) which is canonical with respect to that ordering.*

**Proof.** We give a constructive proof. Let $T$ be a full and fulsome Steiner tree for $N$ with topology $\mathcal{T}$. Since $T$ is full, $\mathcal{T}$ contains $2n-3$ edges (where $n = |N|$) . Assign the natural numbers $1$ to $2n-3$ to the edges of $\mathcal{T}$ to reflect the given ordering of edges. We can use the edge numbers of $\mathcal{T}$ to also number the corresponding edges of $T$, or any Steiner tree for $N$ with topology $\mathcal{T}$. We next show that we can iteratively apply the Complete Zero-Shift Theorem (Theorem 2.19) to transform $T$ to another full Steiner tree for $N$ with the same topology, which we call $T'$, and which is canonical with respect to the given ordering. Suppose, on the contrary, there is an edge $e_1$ with an exclusively secondary segment whose label is strictly less than that of an edge $e_2$ with an exclusively primary segment. Then, by Theorem 2.19, there exists a complete zero-shift for $e_1$ and $e_2$ on the path between these two edges. After applying this shift we either have that $e_1$ has no exclusively secondary segment or $e_2$ has no exclusively primary segment. Since a complete zero-shift does not change the direction of any other straight edges of $T$, the shift reduces the number of primary/secondary edge pairs where the primary edge has a higher number than the secondary edge. Hence, by repeatedly applying zero-shifts, we can construct a tree $T'$, with the same length as $T$, such that no primary edge of $T'$ has a higher number than any secondary edge. ∎

Depending on the chosen ordering, various canonical forms can be obtained. From an algorithmic point of view, the *depth-first ordering* is the most important one. Set some terminal $r$ as the root of $\mathcal{T}$ and order the edges as they appear in a depth-first traversal of the tree from $r$. This results in a canonical form that is illustrated in Figure 2.17. If we divide such a canonical tree $T$ into two subtrees $T_1$ and $T_2$ by deleting the transition edge, the tree $T_2$ that does not contain the root is secondary (i.e., all edges in $T_2$ are secondary edges). Furthermore, the path $\mathcal{P}$ from $r$ to the transition edge is primary, while the subtrees that are attached to $\mathcal{P}$ are each either primary or secondary subtrees. These properties are used by the linear-time algorithm for constructing a full Steiner tree for a given full Steiner topology (Section 2.4) and by the GeoSteiner algorithm (Section 2.6).

By choosing another ordering of the edges, we obtain the following interesting theorem [70]:

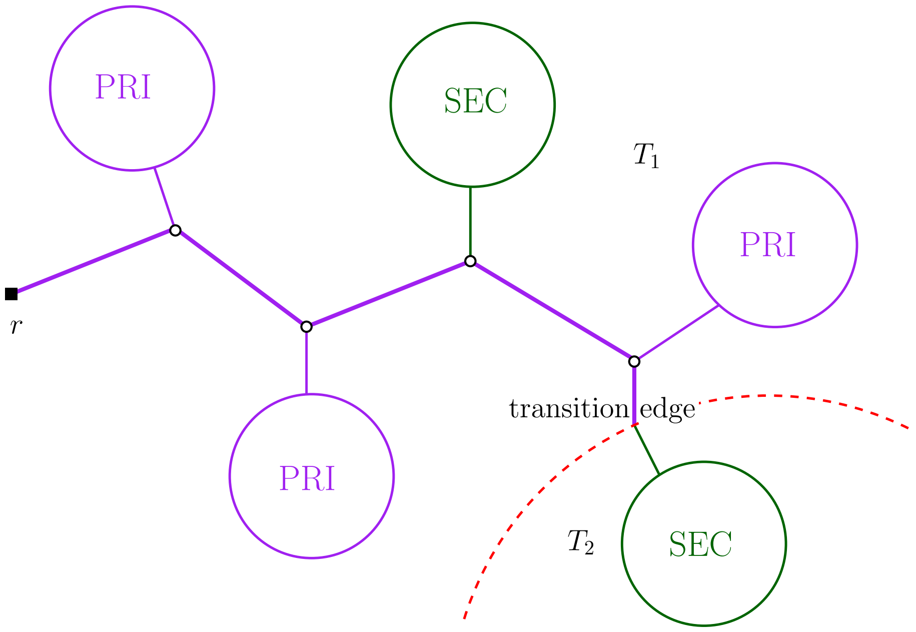


Figure 2.17: Illustration of a tree $T$ with the depth-first order canonical form. Primary edges are drawn in purple, secondary edges are drawn in dark green. The dashed red curve indicates the division of $T$ into two subtrees, $T_1$ and $T_2$.

**Theorem 2.24** *Let $T$ be a full and fulsome fixed orientation Steiner tree for a terminal set $N$, with topology $\mathcal{T}$. Then there exists a full fixed orientation Steiner tree $T'$ for $N$ with topology $\mathcal{T}$ that uses at most* four *legal orientations.*

**Proof.** Order the edges of $T$ by their *colour*; for example, the red edges may come first in the ordering followed by the green edges and then the blue edges. This is a legitimate method of ordering, since edge colours are invariant under zero-shifts. Consider a canonical full Steiner tree $T'$ that comes from this ordering (by Theorem 2.23 such a tree exists). Then it follows from the canonical form that the edges of any given colour, not the same as the colour of the transition edge, are either all primary or all secondary. Hence, $T'$ uses at most four legal orientations. ∎

In Chapter 3 we will discuss the relationship between the well-known Hwang canonical form for *rectilinear* trees [211, 429] and the canonical forms developed in this section.

## 2.4 Algorithms for a given topology

In this section we discuss algorithms for efficiently constructing and understanding the properties of fixed orientation Steiner trees, i.e., fixed orientation interconnection networks that are minimum with respect to their terminals for a given Steiner topology. As with the Euclidean Steiner tree problem, if the topology is known, then the minimum tree can be constructed in polynomial time — here, indeed, in linear time. We begin by giving a straightforward linear programming formulation of the problem, which immediately shows that the problem can be solved in polynomial time. We then take advantage of the canonical form developed in the previous section to first give a very simple quadratic-time algorithm, and then a linear-time algorithm for the construction.

These methods only construct a single fixed orientation Steiner tree for a given topology. Generally, however, a continuum of such trees (all of the same length) exists, due to the fact that the metric is convex but not strictly convex. In the third subsection, we show how to efficiently construct the so-called *flexibility polygon* for a given topology, which shows the extent to which Steiner points and edges of the Steiner tree can move within this continuous family of minimal trees.

### 2.4.1 Linear programming formulation

Linear programming gives a conceptually straightforward way of showing that the fixed orientation Steiner tree problem for a given topology can be solved in polynomial time. The fact that the unit circle $\mathcal{C}$ for the weighted fixed orientation metric is linearly constrained (and convex) makes it possible to compute distances using linear programming — and hence to compute fixed orientation Steiner trees for any given topology by solving a linear programming problem.

Our discussion here is based on the approach of Zachariasen [432], which corrected and reformulated an argument of Xue and Thulasiraman [419].[31]

As before, let $\mathbf{u}_l$, $l = 0, \ldots, 2\sigma - 1$, be the $2\sigma$ vectors that define the extreme points of the unit circle $\mathcal{C}$ (in counter-clockwise order around the circle). The successor of unit vector $\mathbf{u}_l$ is the vector $\mathbf{u}_{l+1}$, where $\mathbf{u}_{l+1} = \mathbf{u}_0$ when $l = 2\sigma - 1$. Suppose we are given two points $p$ and $q$, expressed in Cartesian coordinates, and we wish to compute the distance $\|pq\|$ under the metric given by $\mathcal{C}$. Let $\{\alpha_l, \beta_l\}$ be the unique solution to

$$q = p + \alpha_l \mathbf{u}_l + \beta_l \mathbf{u}_{l+1}$$

for each $l = 0, \ldots, 2\sigma - 1$. A straightforward argument (Exercise 2.11) shows that

$$\|pq\| \; = \max_{l \in \{0,\ldots,2\sigma-1\}} (\alpha_l + \beta_l)$$

and therefore that $d_{pq} = \|pq\|$ can be computed by solving the following linear program:

$$\begin{array}{ll} \text{minimise} & d_{pq} \\ \text{subject to} & \alpha_l + \beta_l \;\; \leq \;\; d_{pq}, \quad l \in \{0, \ldots, 2\sigma - 1\}. \end{array}$$

Note that $\alpha_l$ and $\beta_l$ depend linearly on the coordinates of $p$ and $q$. By applying the constraints in the above formulation for each of the edges in the given topology, a Steiner tree for this topology can be computed in polynomial time (in the size of the input). Clearly, degenerate and non-tree topologies can also be handled by this formulation.

A special case of the above problem, the (separable) rectilinear problem [76], is known to be the dual of a transshipment problem. Similar connections are currently unknown for the general problem.

## 2.4.2 Algorithms based on the canonical form

We now present two algorithms for constructing a fixed orientation Steiner tree for a given terminal set $N$ and a given full Steiner topology $\mathcal{T}$. Both algorithms exploit the canonical form developed in Section 2.3.4, and both depend strongly on a so-called merging operation that can be performed in constant time. The first simple algorithm that we present runs in $O(\sigma n^2)$ time (where, as before, $2\sigma$ is the number of extreme points of $\mathcal{C}$ and $n$ is the number of terminals), while the second runs in $O(\sigma n)$ time — which is the best time possible for a given $\mathcal{C}$.

### Constant-time merging operation

The following lemma establishes conditions under which we can uniquely construct a Steiner point adjacent to two given points, given a knowledge of the direction set and whether incident edges are primary or secondary. The operation of constructing such a Steiner point and replacing the two given points by this Steiner point will be referred to as a *merging* operation. The purpose of this operation is to facilitate the efficient bottom-up construction of the tree, based on the canonical form.

**Lemma 2.25** ***[Constant-time merging]*** *Let $T$ be a fixed orientation Steiner tree with full Steiner topology $\mathcal{T}$, and let $s$ be a Steiner point in $\mathcal{T}$. Assume that the locations in $T$ of two of the neighbours of $s$, $u$ and $v$ are known; furthermore, assume that each edge $(s, u)$ and $(s, v)$ is straight and has been labelled primary or secondary. Finally, assume that we know the direction set for $T$. If the Steiner point $s$ exists in $T$ and is not collinear with $u$ and $v$, then its location is unique and can be computed in constant time; also, the colours of the edges $(s, u)$ and $(s, v)$ — and hence also the colour of the third edge incident with $s$ — are unique.*

**Proof.** For the given direction set, we first make the following simple observations:

1. An exclusively primary edge and an exclusively secondary edge of the same colour use adjacent orientations;

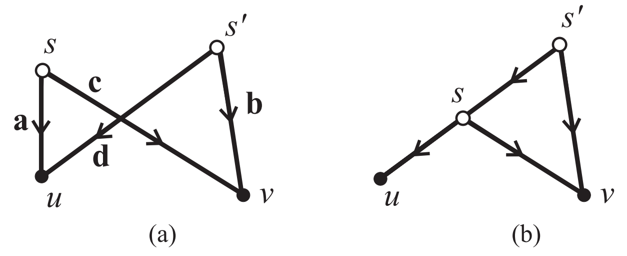


Figure 2.18: Diagrams (a) and (b) illustrate Cases 1 and 2, respectively, in the proof of Lemma 2.25.

2. Every meeting angle is at most $\pi$.

Now suppose, contrary to the statement of the lemma, that it is possible to construct two distinct Steiner points $s$ and $s'$ adjacent to both $u$ and $v$. Direct each incident edge so that it is pointing outwards from $s$ or $s'$. These incident edges use four (not necessarily distinct) directions, all of which belong to a single direction set. We will consider two cases, each of which show that we reach a contradiction to one of the observations above.

**Case 1**. Suppose that $s$ and $s'$ are not collinear with $u$ or with $v$.

We will first show that the interiors of two of the edges intersect at a single point. Consider the colours of the four edges $(s,u)$, $(s,v)$, $(s',u)$ and $(s',v)$. Since there are only three colours, at least one colour (say, red) appears twice. Without loss of generality, we can assume that $(s,u)$ is a red edge. Then $(s',v)$ must also be a red edge (since the other edge incident with $u$ has the same primary/secondary labelling as $(s,u)$ but is not collinear, and the two edges incident with $s$ have distinct colours).

We introduce the following vectors (see Figure 2.18(a)):

$$\mathbf{a} = \overrightarrow{su},\ \mathbf{b} = \overrightarrow{s'v},\ \mathbf{c} = \overrightarrow{sv},\ \mathbf{d} = \overrightarrow{s'u}.$$

Assume the two red edges $(s,u)$ and $(s',v)$ do not intersect. We will show that the other two edges $(s,v)$ and $(s',u)$ do intersect. Note that this occurs if and only if there exist real numbers $0 < k_1 < 1$ and $0 < k_2 < 1$ satisfying the following condition:

$$\mathbf{a} = k_1\mathbf{c} + k_2\mathbf{d}. \tag{2.3}$$

Also note that $\mathbf{a} - \mathbf{d} = \mathbf{c} - \mathbf{b}$, that is:

$$\mathbf{a} + \mathbf{b} = \mathbf{c} + \mathbf{d}. \tag{2.4}$$

First suppose $(s, u)$ and $(s', v)$ both use the same red direction, that is, $\mathbf{b} = k\mathbf{a}$ for some $k > 0$. From Equation (2.4) we obtain: $\mathbf{a} = (\mathbf{c} + \mathbf{d})/(1 + k)$, which satisfies condition (2.3) where $k_1 = k_2 = 1/(k + 1)$.

On the other hand, suppose that $(s, u)$ and $(s', v)$ use different red directions (one exclusively primary and the other exclusively secondary). By observation 1, these are adjacent orientations; hence, $\mathbf{a}$ and $\mathbf{b}$ are linearly independent. This means there exist $\alpha_1, \alpha_2, \beta_1, \beta_2 \in \mathbb{R}$ such that $\mathbf{c} = \alpha_1\mathbf{a} + \alpha_2\mathbf{b}$ and $\mathbf{d} = \beta_1\mathbf{a} + \beta_2\mathbf{b}$. Since $\mathbf{c}$ and $\mathbf{d}$ do not use red directions, at least one of the $\alpha_i$'s and at least one of the $\beta_i$'s must be negative. Again it can be shown that $(s, v)$ and $(s', u)$ intersect (see Exercise 2.12).

We have now established that the interiors of two of the edges intersect at a single point. We can assume that this pair of intersecting edges is $(s, v)$ and $(s', u)$ (noting that we are no longer carrying across any of the previous assumptions about colours of edges). This is the situation shown in Figure 2.18(a).

If $(s, v)$ and $(s', u)$ are the same colour, then the orientation corresponding to the direction of $(s, u)$ lies strictly between the directions of $(s, v)$ and $(s', u)$, contradicting observation 1. On the other hand, if $(s, v)$ and $(s', u)$ have different colours, then the remaining two edges must both use the third colour (since the primary and secondary labelling of each edge is known). Hence, the four edges $(s, u)$, $(s, v)$, $(s', u)$ and $(s', v)$ use three distinct colours. However, the four vertices of the polygonal unit circle $\partial\mathcal{D}$ (centred at the origin) corresponding to the directions of these four edges all lie in one of the open half-planes on one side of the line through the origin with direction $\overrightarrow{ss'}$. This means that one of the meeting angles is greater than $\pi$, contradicting observation 2.

**Case 2**. The remaining possibility is that $s$ and $s'$ are both collinear with exactly one of the points $u$ and $v$, say $u$ (as in Figure 2.18(b)).

Since $(s, v)$ and $(s', v)$ are either both primary or both secondary, it follows that the two edges have different colours, while the edge $(s', u)$ must have the third colour. As in Case 2, these three directions all lie in an open half-plane, again contradicting observation 2.

We conclude that at most one Steiner point can be constructed.

Note that this construction can be done in constant time since the direction set is known. ∎

## A simple quadratic-time algorithm

Suppose we are given a set of $n$ terminals $N$, a full Steiner topology $\mathcal{T}$ for the set of terminals, and a direction set $D$. The aim is to construct a full fixed orientation Steiner tree $T$ for $N$ with topology $\mathcal{T}$ and direction set $D$ (or prove that no such tree exists).

First choose an ordering of the $2n-3$ edges in $T$. This ordering can be completely arbitrary, and it is clearly possible to find an ordering in $O(n)$ time given any sparse graph representation of $T$. The aim is to construct (if possible) a full Steiner tree having the canonical form given by this ordering. Recall, from the previous section, that such a tree has a unique bent edge, or *transition edge* under the given ordering of the edges. Suppose that the transition edge has number $k$ under the chosen ordering, where $1 \leq k \leq 2n-3$, and that the value of $k$ is known to us. Then, in $O(n)$ time, we can construct $T$ as follows. Label all edges numbered less than $k$ as primary edges and label all edges numbered greater than $k$ as secondary edges. We treat the transition edge as the root of $T$ and iteratively apply the merging operation to leaf nodes sharing a parent until the locations of the endpoints of the transition edge have been constructed. Since the primary/secondary labelling of every edge except the transition edge is known, every merging operation has a unique solution (unless the two edges to be merged are collinear) and can be performed in constant time by Lemma 2.25.

If the two edges to be merged, say $(s,u)$ and $(s,v)$, are collinear, then again the colours of the edges $(s,u)$, $(s,v)$ and the third edge incident with $s$ are uniquely determined (by the proof of Lemma 2.25), but the position of $s$ is not unique. This does not cause a problem in the bottom-up construction algorithm due to Lemma 1.38. As a consequence of this lemma, at most one merging operation in the bottom-up construction algorithm can result in a non-unique Steiner point $s$. In fact, this must then be the final merging operation where the third edge incident to $s$ (i.e., the edge other than $(s,u)$ and $(s,v)$) is the *transition* edge, and the other end of this edge is a terminal $t$. This holds since the construction is moving in towards the transition edge. What we can do here is leave the position of the Steiner point undetermined and construct the transition edge by finding the shortest edge connecting $t$ to the line segment $uv$. Clearly, this can also be done in constant time like an ordinary merging operation.

Since there are $2n-3$ different choices for the transition edge, iterating over all possible combinations and using the $O(n)$ algorithm just described will clearly construct a full and fulsome Steiner tree for $N$, $\mathcal{T}$ and $D$ (if any such tree exists) in time $O(n^2)$. Finally, there are only a constant number of possible direction sets (up to $2\sigma$), all of which can be efficiently generated by Theorem 2.10 — in fact, the discussion immediately following Theorem 2.10 shows that all directions can be generated in time $O(\sigma)$. The result is summarised in the following theorem:

**Theorem 2.26** *Let a set of $n$ terminals $N$ and a full Steiner topology $\mathcal{T}$ for that set of terminals be given. Then in $O(\sigma n^2)$ time we can either construct a full and fulsome fixed orientation Steiner tree for $N$ with topology $\mathcal{T}$, or determine that no such tree exists.*

**A linear-time algorithm**

We conclude this section by outlining an argument showing that the above quadratic algorithm can be improved to run in linear time by a careful choice of the order in which the merging operations take place. Here we simply sketch the strategy. Full details for the $\lambda$-geometry case can be found in [70] — the arguments there easily generalise to all fixed orientation Steiner trees.

The idea of the linear-time algorithm (for a given topology $\mathcal{T}$ and fixed direction set $\mathcal{D}$) is that if there exists a full and fulsome fixed orientation Steiner tree $T$ with full Steiner topology $\mathcal{T}$ and direction set $\mathcal{D}$, then $T$ can be assumed to have the depth-first order canonical form illustrated in Figure 2.17. The new algorithm consists of two depth-first traversals of $T$ from an arbitrary root terminal $r$:

1. For the first traversal we assume that all edges are labelled as secondary edges. Secondary subtrees (that is, subtrees where all edges are secondary edges) are constructed bottom-up in $T$. The merging operation is performed bottom-up for each Steiner node; if a Steiner point cannot be constructed, then none of the ancestors of the node (relative to $r$) can be constructed either.

2. For the second traversal, primary subtrees are constructed bottom-up in $T$. At the same time, each edge in $T$ is tried as a potential transition edge. The idea is that it is possible iteratively to construct the endpoints of the transition edge in constant time per edge: the endpoint that is closest to the root $r$ is constructed in the second traversal, while the other endpoint is constructed in the first traversal of $T$ (see Figure 2.17).

Full implementation details are given by Brazil et al. [70]. Thus, we have the following theorem (from [74]):

**Theorem 2.27** *Let a set of $n$ terminals $N$ and a full Steiner topology $\mathcal{T}$ for that set of terminals be given. Then in $O(\sigma n)$ time we can either construct a full and fulsome fixed orientation Steiner tree for $N$ with topology $\mathcal{T}$, or determine that no such tree exists.*

### 2.4.3 Algorithms for flexibility polygons

Minimum Steiner trees in fixed orientation metrics are usually not unique. Non-unique minimum Steiner trees are *flexible* in the sense that we may choose amongst several (lengthwise equally good) embeddings of these minimum Steiner trees. Hence, flexibility is a measure of the extent to which edges and Steiner points in the minimum length network can be perturbed without increasing the length of the network. This has important

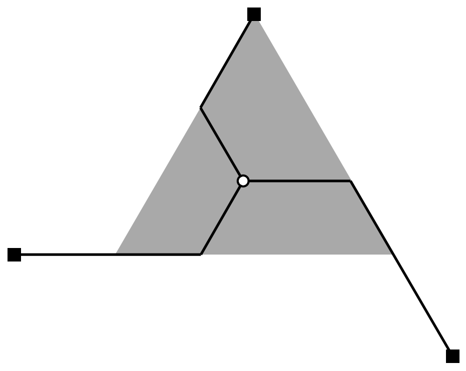


Figure 2.19: Triangular flexibility polygon for three terminals and $\lambda = 3$. The Steiner point may be placed anywhere in the grey-shaded region.

applications in solving multi-objective optimisation problems in chip design, involving minimisation of negative effects of properties such as congestion or signal delay as a secondary objective [41, 42, 307].

In this section we characterise flexibility formally by defining the *flexibility polygon* for a given topology (and for each of the Steiner points in this topology). This concept was introduced by Brazil, Winter and Zachariasen [72, 73], and they furthermore gave an efficient algorithm to construct the flexibility polygon. The original algorithm [73] was only given for $\lambda$-geometry, but it can easily be generalised to arbitrary weighted fixed orientation metrics by applying the structural results from Brazil and Zachariasen [74].

The results related to the flexibility polygon subsume some of the previous work on flexibility. One of the earliest papers in this area was that of Yan et al. [420] which considered the problem of constructing a minimum Steiner tree with three terminals and one Steiner point for the $\lambda = 3$ case, and showed that the set of feasible Steiner points forms a region bounded by an equilateral triangle (Figure 2.19).[32] (See, also, Exercise 2.15.)

More generally, the feasible region for a Steiner point in a minimum fixed orientation Steiner tree is a convex polygon with up to six vertices. This is not surprising given that the problem of constructing a minimum fixed orientation Steiner tree for a given topology can be solved by linear programming, where the coordinates of Steiner points are variables [419, 432]. Even if linear programming in principle can be used to construct feasible Steiner point regions, this would not be as efficient as direct computation, as is outlined in this section.

**Definition [Flexibility polygon]**: For a set of terminals $N$ and a full Steiner topology $\mathcal{T}$ for $N$, we denote by $S(N, \mathcal{T})$ the *set* of all full and fulsome minimum Steiner trees interconnecting $N$ with topology $\mathcal{T}$. The *flexibility polygon* for a terminal set $N$ and a full Steiner topology $\mathcal{T}$ is defined to be the *union* of the embeddings of all minimum Steiner trees in $S(N, \mathcal{T})$.

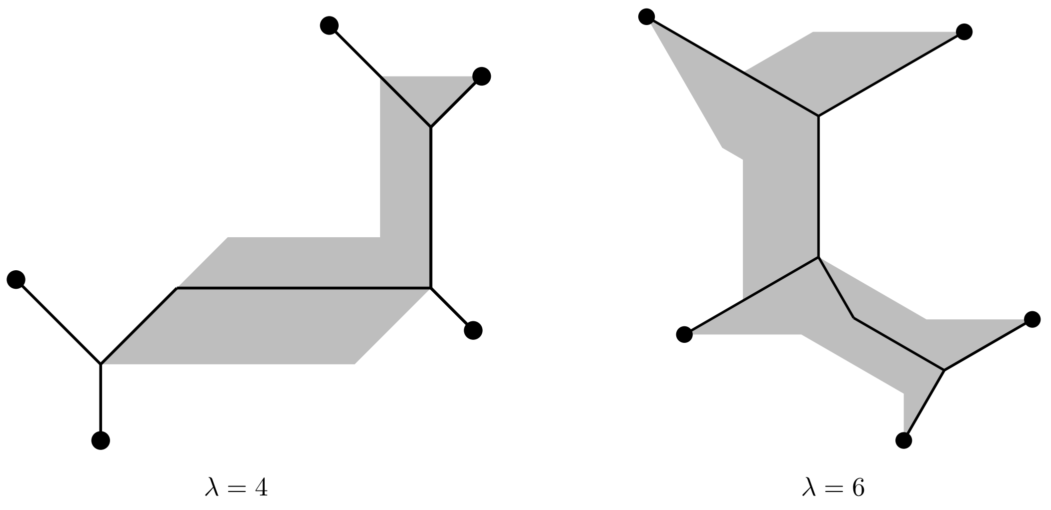


Figure 2.20: Examples of $\lambda$-minimum Steiner trees and flexibility polygons. Notice that a flexibility polygon may have overlapping boundary segments, indicating that parts of some edges may exhibit no flexibility at all.

It can be shown that this union forms a simply connected region with a polygonal boundary whose vertices include the terminals $N$ [73]. Some examples of flexibility polygons for minimum $\lambda$-geometry Steiner trees for the cases where $\lambda = 4$ and $\lambda = 6$ are shown in Figure 2.20.

In the remainder of this section we briefly discuss a constructive algorithm that gives the following theorem of Brazil et al. [73]:

**Theorem 2.28** *Given a full Steiner topology $\mathcal{T}$ with $n$ terminals and a weighted fixed orientation metric with $\sigma$ legal orientations, the flexibility polygon for $\mathcal{T}$ can be computed in $O(\sigma n)$ time.*

The first step of the algorithm is to compute a minimum Steiner tree $T$ by applying Theorem 2.27. The minimum Steiner tree $T$ implicitly identifies a direction set that is used by every Steiner point in $T$. Recall that a direction set consists of directions of three different colours. For each colour in the direction set there are either one or two directions; in the latter case there is one primary and one secondary direction, and in the former case there is a single direction that can be thought of as being both primary and secondary. Note that minimum Steiner trees that use a direction set with 5 or 6 directions usually have much more flexibility than minimum Steiner trees that use a direction set with 4 directions. (As an example, minimum Steiner trees in $\lambda$-geometry where $\lambda$ is a multiple of 3 usually have the greatest flexibility since for these minimum Steiner trees the corresponding direction sets have 6 directions.)

Consider a counter-clockwise outer walk of $T$, beginning and ending at the same terminal. This defines a set of *convex paths* in $T$ that have terminals as endpoints and Steiner points as interior points. In other words, these are paths between terminals where at each intermediate Steiner point the rightmost outgoing edge is taken.

For each convex path we now seek an embedding of the minimum Steiner tree that pushes the path as far as possible to the right (with respect to the direction of the walk along each path), defining a so-called *rightmost* convex path. This means that for each edge $e$ on the rightmost convex path there is no alternative embedding where the same edge is to the right of $e$. It can be shown that the collection of these rightmost convex paths defines the flexibility polygon.

Consider a convex path $P = v_1 v_2 \dots v_{k-1} v_k$ connecting two terminals $v_1$ and $v_k$. We define an ordering of the edges of $\mathcal{T}$ by making a depth-first traversal from $v_1$. At every Steiner point $v_i$, the subtree of $\mathcal{T}$ with its root at $v_i$ (and not intersecting $P$) is traversed before the edge $v_i v_{i+1}$ is traversed. The main technical result is now that the minimum Steiner tree that has the canonical form given by this ordering defines the embedding of the rightmost convex path from $v_1$ to $v_k$ [73].

In order to compute these rightmost convex paths efficiently the algorithm first constructs all primary and secondary subtrees of $\mathcal{T}$, that is, embeddings that consist of primary (or respectively, secondary) directions only. This can be accomplished in $O(n)$ time even though there are $O(n)$ potential subtrees. The algorithm works in a bottom-up fashion by maintaining a queue of subtrees that can be constructed at a given point in time.

The final construction of rightmost convex paths, and hence the flexibility polygon, is achieved essentially by traversing a primary subtree as far as possible along a convex path — and then switching to a secondary subtree with its root at the opposite end of the edge where the primary subtree ends. The construction only requires one outer walk of $T$ and can be performed in constant time for each edge traversed — in total $O(n)$ time.

For a given Steiner point $s$ in $\mathcal{T}$, the union of all feasible positions of $s$ in the minimum Steiner trees in $S(N, \mathcal{T})$ is denoted the flexibility polygon for $s$. Given the flexibility polygon for $\mathcal{T}$, the flexibility polygon for $s$ can be constructed in constant time and has at most six vertices [73]. Examples of such flexibility polygons are shown in Figure 2.21.

## 2.5 Global properties of minimum Steiner trees

The previous sections have focused mainly on fixed orientation Steiner trees, i.e., fixed orientation trees that are minimal for their given Steiner topology. In this section we look at properties relating to globally minimum fixed orientation Steiner trees — in particular, we discuss known bounds on the Steiner ratio, briefly study the generalised Hanan grid

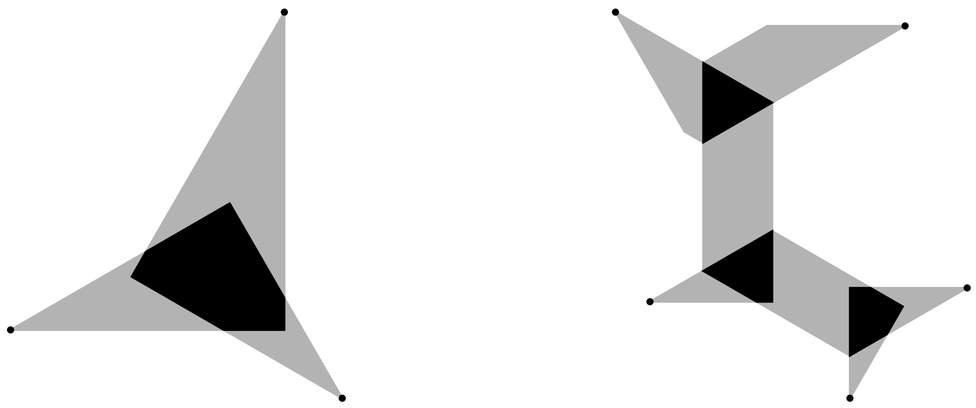


Figure 2.21: Examples of flexibility polygons (dark-shaded) for Steiner points ($\lambda = 6$). The flexibility polygon on the left has 6 vertices, while the flexibility polygons on the right have 3, 4 and 5 vertices, respectively.

reduction, and show that the problem of constructing globally minimum trees is NP-hard.

### 2.5.1 Steiner ratios

Consider the problem of optimally interconnecting a set of terminals $N$ under a metric given by an arbitrary unit circle $\mathcal{C}$ *without* being allowed to use Steiner points. As seen in Chapter 1, this corresponds to computing a *minimum spanning tree (MST)* for $N$ — a problem that is polynomially solvable.

Let $T_{\mathcal{C}}(N)$ and $\overline{T}_{\mathcal{C}}(N)$ denote a minimum Steiner tree and a minimum spanning tree, respectively, for $N$ under the metric given by $\mathcal{C}$. As in Section 1.3.1, define

$$\rho_{\mathcal{C}}(N) = \frac{\|T_{\mathcal{C}}(N)\|}{\|\overline{T}_{\mathcal{C}}(N)\|}$$

to be the ratio between the length of a minimum Steiner tree and a minimum spanning tree for $N$. The *Steiner ratio* $\rho_{\mathcal{C}}$ for the metric given by unit circle $\mathcal{C}$ is defined as

$$\rho_{\mathcal{C}} = \inf_{N} \rho_{\mathcal{C}}(N).$$

In other words, the Steiner ratio is the smallest possible ratio between the minimum Steiner tree and minimum spanning tree lengths for any set of terminals. If $\mathcal{C}$ is the unit circle for the $\lambda$-geometry metric (for a given $\lambda$), then we denote $\rho_{\mathcal{C}}$ by $\rho_{\lambda}$. In this section we briefly survey the results on the Steiner ratio, mainly for $\lambda$-geometry.

Let $\rho$ denote the Steiner ratio for the Euclidean metric. As discussed in Section 1.3.1, the value of $\rho$ is currently unknown, but is strongly conjectured to be $\sqrt{3}/2 \approx$

$0.8660$. Similarly, in $\lambda$-geometry for most values of $\lambda$ the Steiner ratio is not known; however, bounds, or in some cases precise values, can be given in terms of $\rho$.

For the general Steiner ratio problem in $\lambda$-geometry, Sarrafzadeh and Wong [341] derived the following inequality:

$$\rho_\lambda \geq \rho \cos \frac{\pi}{2\lambda}. \tag{2.5}$$

Let $T(N)$ and $\overline{T}(N)$ denote a Euclidean minimum Steiner tree and a minimum spanning tree, respectively, for $N$. Note that $\cos(\pi/(2\lambda))^{-1}$ is the maximum ratio of the distance between two points in $\lambda$-geometry and Euclidean geometry (Exercise 2.13). Then, for the case where $\mathcal{C}$ is a regular $2\lambda$-gon, we obtain Equation (2.5) as follows:

$$\begin{aligned} \|T_{\mathcal{C}}(N)\| &\geq |T(N)| \\ &\geq \rho|\overline{T}(N)| \\ &\geq \rho \cos \frac{\pi}{2\lambda} \, \|\overline{T}_{\mathcal{C}}(N)\|. \end{aligned}$$

It follows from this lower bound that for $\lambda = 3$ we have that $\rho_3 \geq 3/4$. For $\lambda = 3$ there is also a matching upper bound for a three-terminal example obtained by placing a terminal on every second vertex of the regular hexagon representing the unit circle.

This is one of only a few values of $\lambda$ for which the Steiner ratio is known. In Chapter 3 we prove that the Steiner ratio for the rectilinear metric ($\lambda = 2$) is $2/3$. Table 2.2 summarises the results known for $\lambda$-geometry. It is interesting to note that the Steiner ratio appears not to be a monotonically increasing function of $\lambda$. However, both the lower and upper bounds approach $\rho$ as $\lambda \to \infty$ as could be expected.

| Metric | Steiner ratio | References |
|---|---|---|
| $\lambda = 2$ (rectilinear) | $\frac{2}{3}$ | [211] |
| $\lambda = 3$ (hexagonal) | $\frac{3}{4}$ | [246] |
| $\lambda = 4$ (octilinear) | $\frac{2+\sqrt{2}}{4}$ | [349] |
| $\lambda \equiv 3 \bmod 6$ | $\rho \cos \frac{\pi}{2\lambda}$ | [246] |
| $\lambda \equiv 0 \bmod 6$ | $\rho$ | [246] |
| General $\lambda$ (lower bound) | $\max\{\frac{2}{3}, \rho \cos \frac{\pi}{2\lambda}\}$ | [134, 341] |
| General $\lambda$ (upper bound) | $\min\{\frac{\sqrt{13}-1}{3}, \rho\,(\cos \frac{\pi}{2\lambda})^{-1}\}$ | [109, 134, 341] |

Table 2.2: Overview of the results on the Steiner ratio $\rho_\lambda$ for $\lambda$-geometry.

For general fixed orientation metrics the bounds on the Steiner ratio are the same as those for arbitrary normed planes, since any norm can be approximated arbitrarily closely by a weighted fixed orientation metric. In Minkowski planes, $2/3$ is a tight lower bound on the Steiner ratio [168] (the bound is achieved in, for example, the rectilinear plane — see Chapter 3). The best known upper bound is $(\sqrt{13} - 1)/3 \approx 0.8686$ [134], but it is conjectured that the (tight) upper bound is the Steiner ratio for the Euclidean metric [136].

### 2.5.2 Generalised Hanan grid reduction

For the rectilinear Steiner tree problem it is known that for any terminal set $N$ there exists a minimum Steiner tree in the *Hanan grid*, which is the grid obtained by drawing a horizontal line and a vertical line through each terminal. For more on the properties and role of the Hanan grid, see Section 3.2.2 of Chapter 3. The definition of the Hanan grid can be generalised in a natural way to any fixed orientation metric by defining it as the grid resulting when lines of every legal orientation are drawn through each terminal. Sarrafzadeh and Wong [341], however, have pointed out that when going from two to three orientations in the plane, there exist terminal sets for which every minimum Steiner tree has Steiner points that are *not* in the corresponding Hanan grid.

A natural question is therefore: Does there exist a 'small' grid structure in which a minimum Steiner tree for the general fixed orientation problem can always be found? As for the rectilinear problem, the existence of such a grid — which induces a weighted planar graph — would make it possible to reduce the fixed orientation problem to the Steiner tree problem in graphs. In this section we survey the results relating to such a reduction.

**Definition [Generalised Hanan grid]**: For any set of points $P$, define $\mathbf{GG}(P)$ to be the set of intersection points obtained by drawing lines in all legal orientations through every point in $P$. Define the *generalised Hanan grid* $\mathbf{GG}_i$ as follows: For terminal set $N$, $\mathbf{GG}_0(N) = N$ and for $i > 0$ recursively define $\mathbf{GG}_i(N) = \mathbf{GG}(\mathbf{GG}_{i-1}(N))$. Note that $\mathbf{GG}_1(N)$ coincides with the vertices of a Hanan grid for the rectilinear metric.

A single step in this recursive process is illustrated in Figure 2.22. It is, perhaps, surprising to see how rapidly the density of the grid increases after a single iteration for only four legal orientations.

The generalised Hanan grid was introduced in 1992 by Du and Hwang [137]; these authors also established the first version of the following theorem (for $\lambda$-geometry with $\lambda = 3$). This theorem has had a fairly long history[33] before being stated in the following form.

**Theorem 2.29** *For any polygonal unit circle $\mathcal{C}$ and each set of $n$ terminals $N$ there exists a minimum fixed orientation Steiner tree $T$ for $N$ such that all Steiner points are in $\mathbf{GG}_{n-2}(N)$.*

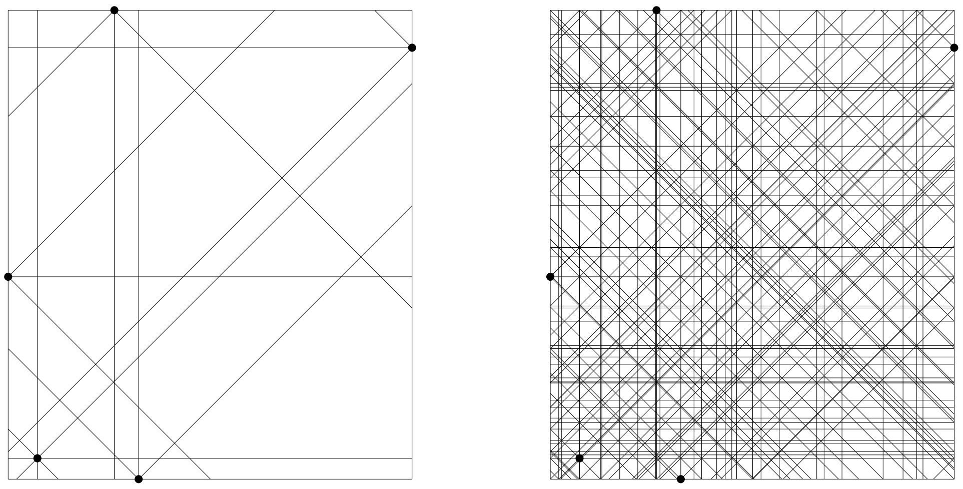


Figure 2.22: Generalised Hanan grid $\mathbf{GG}_1(N)$ (left) and $\mathbf{GG}_2(N)$ (right) for $\lambda = 4$, and a set $N$ with 5 terminals. Only the segments of the lines that are within the rectangular bounding box of the terminals are shown.

**Proof.** We assume that $T$ is a full and fulsome fixed orientation Steiner tree for $T$ (for if it is not, we simply apply the same argument below to each full component). By Corollary 2.20, we can also assume, without loss of generality, that $T$ contains at most one bent edge. The theorem then easily follows since (if $n > 2$) there must exist a Steiner point $s$ in $T$ that is connected with straight edges to *two* terminals; hence, $s$ is in $\mathbf{GG}_1(N)$. By removing the two terminals and their straight edges, and now treating $s$ as a terminal (like the pseudo-terminals in the Euclidean case), the argument can be repeated for all $n - 2$ Steiner points. ∎

The bound provided by Theorem 2.29 can be improved for special cases of the fixed orientation problem. For $\lambda = 3$ (three uniform orientations) a bound of $\lceil (n-2)/2 \rceil$ follows from the fact that for $n \geq 4$ it is possible to perform zero-shifts in such a way that *two* Steiner points simultaneously are connected to two terminals using straight edges only [420]. For three arbitrary orientations ($\sigma = 3$) a weaker bound of $\lceil (n-1)/2 \rceil$ is known [259, 261], and for four uniform orientations ($\lambda = 4$), the best known bound is $\lceil 2n/3 \rceil - 1$ [262, 264]. The bounds for $\lambda = 3$ and $\sigma = 3$ are known to be tight; furthermore, it is known that the bound for $\lambda = 4$ must be strictly greater than $\lceil (n-2)/2 \rceil$ [259, 261].

It has been conjectured that the bound in Theorem 2.29 can be improved [45, 262, 264]. There are nevertheless arguments that support the opposite fact, and we conjecture that the bound in Theorem 2.29 is tight.

From an algorithmic point of view, Theorem 2.29 has limited use since the number of vertices in $\mathbf{GG}_i(N)$ is $\Omega(n^{i+1}\sigma^i)$ for $|N| = n$, which is $\Omega(n^{n-1}\sigma^{n-2})$ for $i = n - 2$. Even for small problem instances, the generalised Hanan grid is too large to be useful (see Figure 2.22).

### 2.5.3 Computational complexity

It seems natural to conjecture that a suitably discretised version of the fixed orientation Steiner tree problem is always NP-hard. More precisely, one can ask whether for a given convex centrally symmetric polygon $\mathcal{C}$ the corresponding fixed orientation Steiner tree problem is NP-hard, after applying a discretisation and scaling to the problem (if necessary) to resolve any computational issues arising from working with exact real arithmetic. Until recently, NP-hardness had been established for only two cases, both of which are examples of $\lambda$-geometry Steiner tree problems. These two cases are the rectilinear metric ($\lambda = 2$) [170] and the octilinear metric ($\lambda = 4$) [291]. Here we extend these results to all $\lambda$-geometry Steiner tree problems with $\lambda > 2$, using a restriction of the terminals to parallel lines very similar to that employed in Section 1.3.3 (The rectilinear case, $\lambda = 2$, requires a different approach, which is discussed in Chapter 3.)

The approach here is very similar to that used for the Euclidean case in Chapter 1; we show the problem is NP-hard for a specific set of instances, where the terminals are constrained to lying on two parallel lines. More specifically, we show that the following class of decision problems is NP-complete for each value of $\lambda > 2$.

PARALLEL LINES $\lambda$-GEOMETRY STEINER TREE DECISION PROBLEM
**Instance**: A finite set of points $N$ lying on two parallel lines in the Euclidean plane and a positive integer $L$.
**Question**: Is there a $\lambda$-geometry Steiner tree $T$ with terminal set $N$ such that the length of $T$ is at most $L$?

As in Section 1.3.3, the main strategy of the proof is to show that there is a polynomial tranformation from each instance of the subset sum problem to a corresponding instance of the parallel lines $\lambda$-geometry Steiner tree decision problem. We begin by only considering the case where $\lambda$ is a multiple of 3 (i.e., $\lambda = 3m$ for some positive integer $m$), since the proof of this case is almost identical to the proof of the Euclidean case given in Section 1.3.3. The cases where $\lambda \neq 3m$ pose some additional technical difficulties which we then show how to circumvent.

**Theorem 2.30** *The discretised parallel lines $\lambda$-geometry Steiner tree decision problem is NP-complete for any given $\lambda = 3m$ (where $m$ is a positive integer).*

**Proof.** The proof here is very similar to the proof of Theorem 1.17. The construction of the base tree $T_x$ is the same as in the Euclidean case, as is $T_v$, with the small change that the geodesics from each $b_i$ to $c_i$ may not be straight line segments; in particular, they will be straight line segments when $m$ is even and bent edges when $m$ is odd. This makes a slight change to the value of $L_v$, the length of $T_v$, but not to the validity of the proof. In addition, we now refer to $T_v$ as a *three-orientation tree* (rather than a horizontal tree), indicating that the main full component uses only three legal orientations.

Claims 1 and 2 are essentially identical in their statements and proofs to those in the proof of Theorem 1.17. The only minor change occurs in Claim 3:

**Claim 3.** The following three statements are equivalent:

(A) The answer to the given instance of the subset sum problem is ‘yes’.

(B) There exists a three-orientation minimum $\lambda$-geometry Steiner tree on $N_0$ with the same base topology as $T_v$.

(C) There exists a Steiner $\lambda$-tree on $N_0$ with length at most $L_v - \sqrt{3}d$.

**Proof of Claim 3.** The first parts of this claim, showing that (A) $\Leftrightarrow$ (B), and that (B) $\Rightarrow$ (C), are identical in their methods of proof to the arguments in the first three steps of Claim 3. It only remains to show that $\neg$(B) $\Rightarrow$ $\neg$(C). Suppose we have a minimum Steiner tree $T_0$ on $N_0$ with the same base topology as $T_v$, but which is not a three-orientation tree. As discussed in Section 2.3.3, we can assume that there is only one bent edge in $T_0$ (by Corollary 2.20). Furthermore, since $\mathbf{d}(V_1, V_1') \gg D$, we can assume that the bent edge corresponds to one of the horizontal edges of $T_v$ connecting to some triple $a_i, b_i, c_i$ (by Corollary 2.22) and that this edge is embedded with a single corner point so that the horizontal part is incident with the Steiner point furthest from $x_i$. An example of such a bent edge is illustrated by the red edge in Figure 2.23(a). Now, suppose we replace this bent edge by the orthogonal projection of this edge onto the line extending the horizontal segment of the edge. This is shown in Figure 2.23(b). Note that the disconnected end of this new edge lies on the vertical line through the Steiner point closest to $x_i$, since the projection is in the vertical direction. It is easy to see, by the same argument used to show (B) $\Rightarrow$ (C), that the length of the resulting (disconnected) three-orientation $\lambda$-geometry network is again $L_v - \sqrt{3}d$. By the construction, this disconnected tree has length strictly less than that of $T_0$; hence, the length of $T_0$ is greater than $L_v - \sqrt{3}d$, giving the required conclusion to the claim. ∎

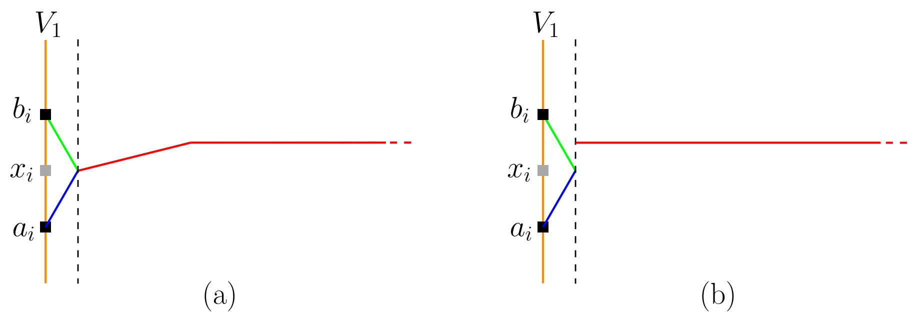


Figure 2.23: Illustration of the construction for the proof of Claim 3. Diagram (a) shows the bent edge closest to $x_i$, while diagram (b) shows the projection of that bent edge onto the line extending its horizontal component. Here the Steiner point is adjacent to $a_i$ and $b_i$ but the same argument and construction apply if the Steiner point is adjacent to $b_i$ and $c_i$. Also, here the non-horizontal component of the bent edge has positive gradient, but again the same argument applies if it has negative gradient, in which case all other edges of $T_0$ are secondary edges.

To conclude the proof of the theorem we need to address the issue of discretisation, which is actually easier here than in the Euclidean case. Recall that in the discretised problem Euclidean distances are rounded up to the nearest integer, and it is assumed that terminals and Steiner points only have integer coordinates. Since all trees considered have at most $7n+1$ edges, every tree is at most length $3 \cdot (7n+1)$ longer or shorter than before the discretisation.

In order to distinguish between ‘yes’ and ‘no’ instances in the discretised problem, we need to be able to distinguish between three-orientation minimum Steiner trees and non three-orientation minimum Steiner trees. Consider a non three-orientation minimum Steiner tree $T_v$. In the ‘worst’ case $T_v$ comes from a subset sum problem for which there exists a subset $J \subseteq \{1, \ldots, n\}$ such that $|d - \sum_{i \in J} d_i| = 1$. From the proof that (B) $\Rightarrow$ (A) in Claim 3 it follows that $T_v$ can be transformed into a three-orientation tree $T'_v$ by moving exactly one $x_i$ upwards or downwards by distance 1. From the last part of the proof of Claim 3 (and Figure 2.23) it follows that the length of $T'_v$ is $\epsilon_\lambda = (1 - \cos\omega)/(\sin\omega)$ greater than $L_v - \sqrt{3}d$. More generally, the length of any non three-orientation minimum Steiner tree is *at least* $\epsilon_\lambda$ greater than $L_v - \sqrt{3}d$.

Suppose we multiply all terminal coordinates of the original problem by some integer $K$ before we perform the discretisation. This increases the minimum length difference to $K\epsilon_\lambda$. By choosing $K$ such that $K\epsilon_\lambda > 2 \cdot 3 \cdot (7n+1)$, we can ensure that a three-orientation minimum Steiner tree is always shorter than a non three-orientation minimum Steiner tree in the discretised problem. Choosing $K > (42n+6)/\epsilon_\lambda$ suffices, and results in a polynomial scaling. ∎

Since every instance of the parallel lines $\lambda$-geometry Steiner tree decision problem is also an instance of the $\lambda$-geometry Steiner tree decision problem, we immediately get the following corollary.

**Corollary 2.31** *The discretised $\lambda$-geometry Steiner tree decision problem is NP-complete for any given $\lambda = 3m$ (where $m$ is a positive integer).*

The proof of Theorem 2.30 relies, to a large extent, on the properties of the base tree $T_x$ constructed in the course of the proof. A key property of the base tree is that if we perturb a single terminal $x_i$ up or down along $V_1$ the resulting minimum Steiner tree on the new terminal set is strictly longer than $T_x$. If $x_i$ is perturbed downwards (away from $v$), then the only change to the tree $T_x$ is that the edge incident with $x_i$ becomes a bent edge via the introduction of a new secondary direction; all other edges in the tree are straight primary edges. On the other hand, if $x_i$ is perturbed upwards (towards $v$), then the edge incident with $x_i$ again becomes a bent edge but this time via the introduction of a new primary direction; all other edges in the tree are straight secondary edges. This is possible due to the symmetry in the direction set for $\lambda = 3m$, which means that in a three-orientation Steiner tree such as $T_x$ it is ambiguous as to whether the edges are all primary or all secondary (see Table 2.1).

The difficulty in generalising Theorem 2.30 to other values of $\lambda$ lies in the fact that the direction sets no longer exhibit this symmetry when $\lambda \neq 3m$. If we construct a base tree for one of these other values of $\lambda$ (as in the proof of Theorem 2.30) using primary directions (as in Table 2.1), then it is no longer true that perturbing $x_i$ in either direction along $V_1$ always reduces the length of the Steiner tree; perturbing $x_i$ upwards again results in the three directions in the base tree all being treated as secondary directions which, because of the lack of symmetry in the direction sets, means that the 'red' edges (i.e., the edges that have more than one direction in the direction set) no longer correspond to the horizontal edges of the base tree. In other words, under the perturbation an edge with a different colour to the edge incident with $x_i$ will become bent. It can be shown that the new minimum Steiner tree that results from perturbing $x_i$ upwards is shorter than the original base tree.

This problem, however, can be successfully circumvented via a slight alteration to the construction, as we show in the following theorem.

**Theorem 2.32** *The discretised parallel lines $\lambda$-geometry Steiner tree decision problem is NP-complete for any given $\lambda > 2$.*

**Proof.** The case where $\lambda = 3m$ is proved in Theorem 2.30, so we can now assume that $\lambda \neq 3m$. As before, let $S = \{d_1, \ldots, d_n\}$ and $d < \sum_{i=1}^{n} d_i := D$ be a given instance of the subset sum problem. The argument here is again similar to the proof of Theorem 1.17

(the Euclidean case), except for a few crucial differences in the construction, which we indicate below.

We again begin by constructing an instance of the parallel lines $\lambda$-geometry Steiner tree decision problem designed to encode the given instance of the subset sum problem. Recalling that $\omega := \pi/\lambda$, let $V_1, V_1', V_2', V_2$ be four parallel lines each with a polar slope of $\pi/2 + \omega/3$, ordered from left to right such that

$$\mathbf{d}(V_1, V_2) \gg \mathbf{d}(V_1, V_1') = \mathbf{d}(V_2', V_2) \gg D.$$

Let $u_0$ be a fixed point on $V_2$, and construct a zigzag path $P$ between $u_0$ and a point on $V_1$ (labelled $v$), such that: $P$ is composed of line segments with alternating polar angles $(4m+2)\omega$ and $(2m+1)\omega$ if $\lambda = 3m+1$, or $(4m+3)\omega$ and $(2m+2)\omega$ if $\lambda = 3m+2$; and $P$ has $2n$ internal vertices lying alternatively on $V_1'$ and $V_2'$. As before, horizontal edges are then extended from each of the internal vertices to points $x_i$ on $V_1$ or $u_i$ on $V_2$ to create the base tree $T_x$, as illustrated in Figure 2.24(a).

This base tree is then used to construct a tree $T_v$ (as in Stage 3 of the proof of Theorem 1.17) by replacing each $x_i$ by the triple $a_i$, $b_i$ and $c_i$, satisfying: $|a_ib_i| = d_i$; $|b_ic_i| = d_i$ so that the main full component of $T_v$ is a three-orientation tree. Note that, unlike the $\lambda = 3m$ case, $x_i$ is not the mid-point of $a_ib_i$; see Figure 2.24(b). As in Claim 1 of the proof of Theorem 1.17, it is straightforward to show that $T_x$ and $T_v$ are unique minimum $\lambda$-geometry Steiner trees for their respective terminal sets. Let $L_v := |T_v|$.

For the final stage of the construction, we define the two constants:

$$K_1 = \frac{2\sin(\pi/3 - \omega/3)\cos(\omega/3)}{\sin(\pi/3 + 2\omega/3)}, \quad K_2 = \frac{\cos(\omega/3)}{\sin(\pi/6 + \omega/3)}$$

and let $v_0$ be the point on $V_1$ below $v$ such that $|v_0v| = K_id$ for the case $\lambda = 3m + i$ (where $i \in \{1, 2\}$). As before, we define $T_0$ to be the minimum $\lambda$-geometry Steiner tree on the same terminals as $T_v$ except where $v$ has been replaced by $v_0$. (Note that if we substitute $\omega = 0$ into the expression for $K_1$ or $K_2$ we obtain 2, which corresponds to the constant in the $\lambda = 3m$ case (where the given parallel lines are vertical).)

The aim now is to use this construction to prove an equivalent statement to Claim 3 in the proof of Theorem 2.30. We first note that in the base tree $T_x$, if we treat $v$ and a point $x_i$ as movable points and perturb $x_i$ upwards along $V_1$ by a distance of $\delta$ while keeping the tree a three-orientation tree, then $v$ moves downwards along $V_1$ by a distance of $K_1\delta$ or $K_2\delta$ for the cases $\lambda = 3m + 1$ and $\lambda = 3m + 2$, respectively. Furthermore, the tree reduces in length by a constant multiple of $\delta$ (independent of the choice of $x_i$), say $C\delta$; a straightforward calculation shows that

$$C = \begin{cases} \dfrac{\cos(\omega/3)}{\sin(\pi/3 + 2\omega/3)} + \dfrac{(K_1 - 1)(\cos(\omega/3) - 1/2)}{\sin(\pi/3 - \omega/3)} & \text{if } \lambda = 3m + 1, \\[2ex] \dfrac{\cos(\omega/3) - 1/2}{\sin(\pi/3 + \omega/3)} + \dfrac{(K_2 - 1)\cos(\omega/3)}{\sin(\pi/3 - 2\omega/3)} & \text{if } \lambda = 3m + 2. \end{cases}$$

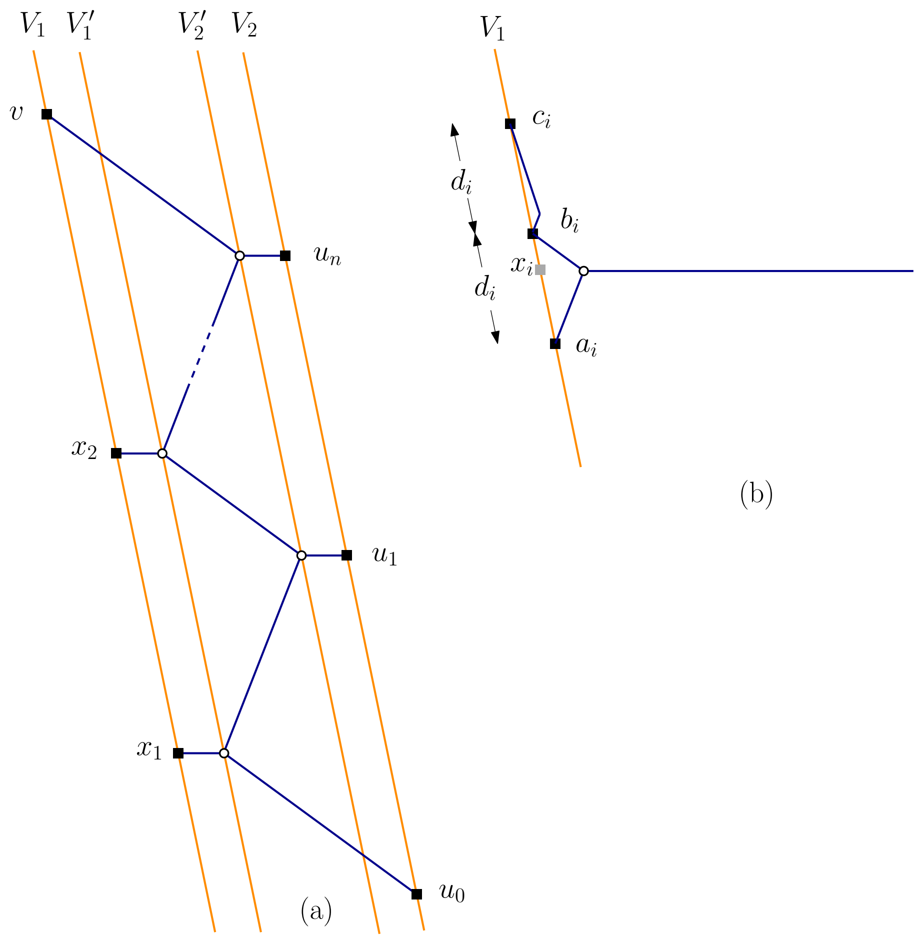


Figure 2.24: Diagram (a) illustrates the base tree $T_x$, for the case $\lambda = 5$. Diagram (b) shows how $T_v$ connects to each triple $a_i$, $b_i$, $c_i$, again for the case $\lambda = 5$.

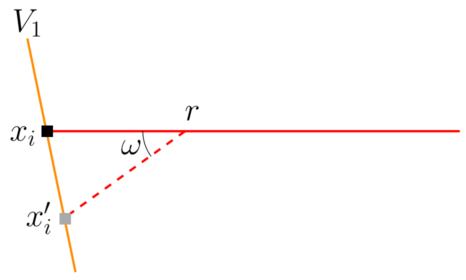


Figure 2.25: The effect of a downward perturbation of $x_i$ on the base tree, illustrated for $\lambda = 5$. Under the perturbation the horizontal edge incident with $x_i$ becomes a bent edge.

(Again note that if we substitute $\omega = 0$ into either expression for $C$ we obtain $\sqrt{3}$, which corresponds to the constant in the $\lambda = 3m$ case.)

Using this notation, we now prove the following claim, equivalent to Claim 3 in the proof of Theorem 2.30.

**Claim.** The following three statements are equivalent:

(A) The answer to the given instance of the subset sum problem is 'yes'.

(B) There exists a three-orientation minimum $\lambda$-geometry Steiner tree on $N_0$ with the same base topology as $T_v$.

(C) There exists a Steiner $\lambda$-tree on $N_0$ with length at most $L_v - Cd$.

**Proof of Claim.** The first parts of this claim, showing that (A) $\Leftrightarrow$ (B), and that (B) $\Rightarrow$ (C), are identical in their methods of proof to the arguments in the first three steps of Claim 3. It only remains to show that $\neg$(B) $\Rightarrow$ $\neg$(C). As before, this statement is equivalent to showing that any perturbation of one of the points $x_i$ in the base tree $T_x$ upwards or downwards along $V_1$ to a new point $x_i'$ creates a fourth direction and strictly increases the length of the tree.

First suppose $x_i'$ lies below $x_i$, as in Figure 2.25.

Then the choice of directions for the non-horizontal edges implies that the red edges include the horizontal direction, and hence we can assume that the edge incident with $x_i'$ becomes bent. We can embed this bent edge so that the corner point $r$ lies on the horizontal line through $x_i$, as in the figure. Since $\angle x_i r x_i' = \omega$ and the polar slope of $V_1$ is $\pi/2 + \omega/3$ it follows that $\angle r x_i x_i' = \pi/2 - \omega/3$ and $\angle r x_i' x_i = \pi/2 - 2\omega/3$; this implies that $|x_i' r| > |x_i r|$ since $x_i' r$ is opposite the larger angle in $\triangle r x_i x_i'$. We can assume that this red edge is the only edge that changes length; hence, the base tree increases in length under the perturbation.

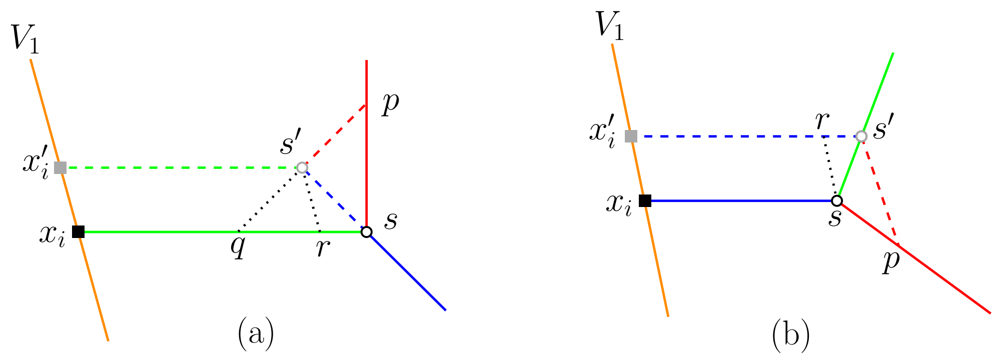


Figure 2.26: The effect of an upward perturbation of $x_i$ on the base tree, illustrated for (a) $\lambda = 4$, and (b) $\lambda = 5$. In (a), under the perturbation $s$ moves to $s'$ and the line segments $ps$ and $x_is$ are replaced by the new edges shown in dashed lines. In (b) the three line segments $ps$, $s's$ and $x_is$ are replaced by the edges indicated in dashed lines.

Next suppose $x_i'$ lies above $x_i$ and $\lambda = 3m + 1$. This is illustrated (for $\lambda = 4$) in Figure 2.26(a).

In this case the red edges are necessarily those that include the polar direction of $(4m + 2)\omega$, which is now the secondary red direction. Let $s$ be the Steiner point adjacent to $x_i$ in $T_x$, and let $s'$ be the location of the Steiner point adjacent to $x_i'$ in the perturbed tree, which we denote $T_x'$. As shown in Figure 2.26(a), under the perturbation of $x_i$ the edge incident to $s$ with polar direction $(2m+1)\omega$ is extended, the horizontal edge incident to $s$ is translated upwards and shortened, while the third edge incident to $s'$ is a bent edge which can be embedded with a single corner point $p$ lying on the third edge incident with $s$. Now let $q$ be the intersection of the line extending $ps'$ with $x_is$, and let $r$ be the intersection of the line through $s'$ parallel to $V_1$ (i.e., with polar direction $\pi/2 + \omega/3$) with $x_is$. A simple calculation shows that $|s'q| = |s's|$.

Define the function $f_1(\omega) = |T_x'| - |T_x|$. Using the notation above and rescaling so that $|ps| = 1$, it follows that

$$\begin{aligned} f_1(\omega) &= |ps'| + |ss'| - |ps| - |sr| \\ &= |pq| - |ps| - |sr| \\ &= 2\cos(\pi/3 - \omega/3) - 1 - \frac{\sin(\omega)(1 - \tan(\pi/3 - \omega/3)\tan(\omega/3))}{2\sin(\pi/3 - \omega/3)}. \end{aligned}$$

If we treat $f_1$ as a continuous function of $\omega$, as graphed on the left in Figure 2.27, then it can be shown analytically that the only root of $f_1$ in the interval $[0, 1]$ is at $\omega = 0$, and consequently that $f_1$ is positive throughout the interior of that interval. Hence, $|T_x'| - |T_x| > 0$ for all relevant values of $\omega$ as required.

Finally, the argument for the case where $x_i'$ lies above $x_i$ and $\lambda = 3m + 2$ is similar

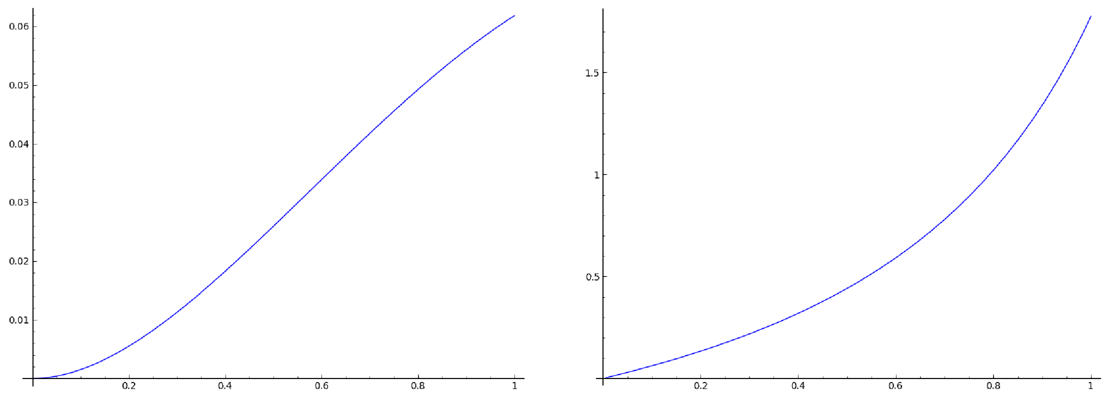


Figure 2.27: The graphs of $f_1$ (left) and $f_2$ (right) as a continuous function of $\omega$ in the interval $[0,1]$.

to the above. In this case it is an edge of $T_x$ with polar direction of $(2m+2)\omega$ that becomes bent, as illustrated in Figure 2.26(b). Again we let $s$ be the Steiner point adjacent to $x_i$ in $T_x$, $s'$ be the Steiner point adjacent to $x_i$ in $T'_x$, and $p$ be the corner point on the bent edge in $T'_x$ (as in the figure). We define $r$ to be the intersection of the line through $s$ parallel to $V_1$ with $x'_i s'$. Define the function $f_2(\omega) = |T'_x| - |T_x|$, and again rescale so that $|ps| = 1$. Then

$$\begin{aligned} f_2(\omega) &= |rs'| + |ps'| - |ss'| - |sp| \\ &= \frac{\sin(\omega)}{2\cos(\omega/3)\sin(2\pi/3-\omega/3)} + \frac{\sin(2\pi/3-\omega/3)-\sin(\omega)}{\sin(\pi/3-2\omega/3)} - 1. \end{aligned}$$

Again, $f_2(\omega)$, as a continuous function, is positive in the interior of the interval $[0,1]$, as verified by the rightmost graph in Figure 2.27.

This concludes the proof of the claim. The issue of discretisation is again easy to deal with (as in the proof of Theorem 2.30), concluding the proof of the theorem. ∎

Theorem 2.32 immediately implies the following corollary.

**Corollary 2.33** *The discretised $\lambda$-geometry Steiner tree decision problem is NP-hard for any given $\lambda > 2$.*

## 2.6 GeoSteiner algorithm

The GeoSteiner approach (see Section 1.4 in Chapter 1) has been adapted to the uniform orientation ($\lambda$-geometry) Steiner tree problem by Nielsen, Winter and Zachariasen [297].

The efficiency of the approach depends critically on the use of canonical forms for fixed orientation full Steiner trees (FSTs) as described in Section 2.3.4.

Independently of the work of Nielsen et al. [297], Coulston [122] has implemented a similar algorithm for $\lambda = 4$. Coulston's algorithm can only solve small problem instances — mainly due to the absence of of the more sophisticated pruning techniques, such as pruning tests based on canonical forms. More recently, Pagh [301] has adapted the GeoSteiner approach to the general weighted fixed orientation metric. Here again no pruning tests based on canonical forms are employed, and consequently this algorithm also does not scale well.

In this section we describe the main ideas of the FST generation algorithm proposed by Nielsen et al. [297]. As noted in Chapter 1, the FST concatenation phase of the algorithm is independent of the underlying metric, and can be reduced to either the Steiner tree problem in graphs (Section 5.1 in Chapter 5) or the minimum spanning tree problem in hypergraphs (Section 5.2.1 in Chapter 5).

## 2.6.1 Top-level FST generation algorithm

Consider a fulsome FST $T$ in a minimum fixed orientation Steiner tree. As in the previous sections, we view $T$ as a tree embedded in the Euclidean plane composed of line segments in legal directions. We can assume that $T$ has at most one bent edge $pq$ (Corollary 2.20), and that $pq$ contains a single corner point $c$; if $T$ consists of straight edges only, let $c$ be the midpoint of an arbitrary edge $pq$ in $T$. Recall the definition of *branches* and *branch trees* from Section 1.4.2. If we cut edge $pq$ at $c$, we obtain two branch trees having *straight edges only*: one rooted at $p$ having a stem (or ray) leaving $p$ along $pc$, and another rooted at $q$ having a stem (or ray) leaving $q$ along $qc$. A *branch* is a set of branch trees that span a common set of terminals and that have a common topology. For the fixed orientation problem, it is convenient to let each branch contain exactly one branch tree; that is, branches and branch trees are identical in the discussion that follows.

As in the Euclidean problem, it is easy to see that we can combine branches/branch trees to form larger branches/branch trees. Branch trees of size 1 consist of a single terminal having a stem leaving in one of the legal directions; hence there are exactly $2\sigma n$ branch trees of size 1 (where $\sigma$ is the number of legal orientations and $n$ is the number of terminals).

Consider two branch trees $B_1$ and $B_2$ with stems rooted at $s_1$ and $s_2$, respectively. Assume that the stems of $B_1$ and $B_2$ intersect at a point $s$. If the directions of the two stems correspond to the directions of two different colours in a single direction set, then we can now form a larger branch tree $B$ having its root at the intersection $s$ — with a new stem that emanates from $s$ (Figure 2.28). Also, if the directions of the two stems correspond to directions of the same colour in a direction set, we can view the intersection $s$ as a potential corner point in a new FST, with $s_1 s_2$ being the single bent edge in the tree. With this interpretation, Algorithm 1.3 from Section 1.4 in Chapter 1 can be applied directly to the minimum fixed orientation Steiner tree problem.

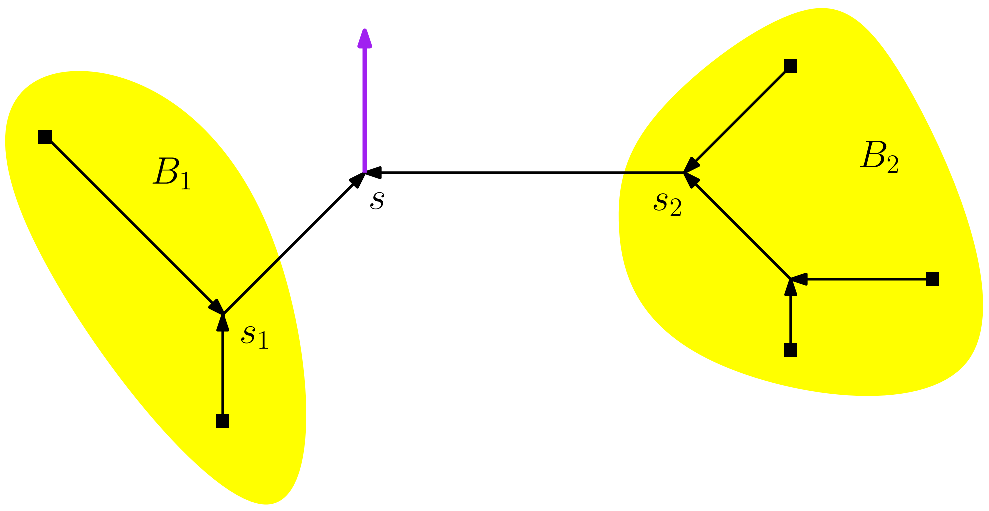


Figure 2.28: Branch trees $B_1$ and $B_2$ with roots $s_1$ and $s_2$ are joined to form a larger branch tree with root $s$. The arrows indicate the paths from the terminals to the root of each branch tree. The purple arrow shows the direction of the new stem.

One of the main tasks of the FST generation algorithm is to prune branches and full Steiner trees (lines 14 and 19 in Algorithm 1.3). In the following section we present the pruning tests applied in [297].

### 2.6.2 Pruning of branch trees and full Steiner trees

In this section we describe the most important tests for pruning branch trees and for pruning FSTs. The tests are presented in the order in which they are used in the algorithm — from the simple tests to the more time-consuming powerful tests. The section concludes with some remarks on the experimental performance of the algorithm.

#### Direction sets

One of the most basic properties of an FST $T$ is that the directions of the edges in $T$ must come from a common direction set. Let $\mathcal{D}_1$ be the collection of direction sets that the edges in branch tree $B_1$ can come from; define $\mathcal{D}_2$ similarly for branch tree $B_2$. Now we must have $\mathcal{D}_1 \cap \mathcal{D}_2 \neq \emptyset$, otherwise the merging of $B_1$ and $B_2$ would result in a non-minimal branch tree. Furthermore, the direction of the new stem emanating from root $s$

must come from a direction set in $\mathcal{D}_1 \cap \mathcal{D}_2$, such that it forms a valid set of directions around $s$. In the algorithm, we create one branch tree for each possible direction of the stem — the number of such stems is at most three (Exercise 2.14).

### Canonical forms

The concept of canonical forms facilitates a pruning procedure based on a strategic choice of canonical form for each FST $T$. The idea is to choose a canonical form that can be tested bottom-up in the algorithm, that is, which has an effect even for small branch trees. This can be achieved by using the edge ordering that comes from a depth-first traversal starting in the lowest indexed terminal of $T$ (under any given ordering of the terminals). To appreciate the effectiveness of such a canonical form, consider two branch trees $B_1$ and $B_2$. Either the lowest indexed terminal is amongst the terminals spanned by $B_1$ and $B_2$, or it is 'outside' (and thus unknown). In the former case, we know which terminal has the lowest index, and this restricts the distribution of directions in the merged branch tree. In the latter case, even more restrictions can be enforced; if neither the restrictions in the former nor the latter case can be fulfilled, the resulting branch tree can be pruned.

More specifically, we use the following canonical form for each FST $T$. Consider any embedding for $T$. Let $t$ be the lowest index terminal in $T$, and number the edges according to a depth-first traversal of $T$ from $t$. The children of a Steiner point in $T$ are visited by the depth-first traversal in the order given by their geometric locations in the embedding of $T$ such that the *leftmost child is visited before the rightmost child* (see Figure 2.17); note that zero-shifts preserve this ordering.

Define a branch tree to be *clean* if the edges use at most three directions, and these three directions form a valid set of purely primary and/or purely secondary edges in some direction set. (For example, for the uniform orientation problem with $\lambda = 3m$ this corresponds to a branch tree where the edges meet at an angle of $2\pi/3$ at every Steiner point.) A branch tree that is not clean is called *mixed*. The following powerful constant-time tests can now be applied:

1. A mixed branch tree can only be merged with a clean branch tree, and the minimum index terminal must reside in the mixed branch tree.

2. When merging a mixed branch tree with a clean branch tree, the directions used by the clean branch tree must correspond to primary (respectively secondary) edges in the mixed branch tree when the clean branch tree is on the left (respectively right) as seen from the mixed branch tree; in addition, only one stem direction is possible, corresponding to a primary edge.

3. When merging two clean branch trees to create a mixed branch tree, the merged branch tree must have the required canonical form; for example, there can be no

primary edge in a right subtree on the path from the minimum index terminal to the new root.

### Other pruning tests

The other pruning tests used in FST generation for the minimum fixed orientation Steiner tree problem are similar to those described in Chapter 1 for the Euclidean case, and hence are only outlined very briefly here.

**Lune property.** Recall that a lune $\mathcal{L}(u, v)$ is defined as the set of points that are strictly within distance $|uv|$ of both $u$ and $v$ (where distance here is given by the fixed orientation metric). If $uv$ is an edge in a minimum fixed orientation Steiner tree, then $\mathcal{L}(u, v)$ cannot contain any terminal (Lemma 1.13). This test can be applied to the two new edges $s_1s$ and $s_2s$ (see Figure 2.28): if a terminal is inside one of the lunes $\mathcal{L}(s_1, s)$ or $\mathcal{L}(s_2, s)$, the merged branch tree cannot be used to generate a full component of a minimum Steiner tree, and hence can be discarded.

**Bottleneck Steiner distance bound.** The bottleneck Steiner distance $\mathrm{BSD}(t_1, t_2)$ bounds the length of each edge on a path between terminals $t_1$ and $t_2$ in a minimum fixed orientation Steiner tree (Lemma 1.14 in Chapter 1). Let $N(B_1)$ and $N(B_2)$ be the set of terminals spanned by branch trees $B_1$ and $B_2$, respectively. Let

$$B = \min_{t_1 \in N(B_1),\ t_2 \in N(B_2)} \mathrm{BSD}(t_1, t_2)$$

be the minimum pairwise bottleneck Steiner distance between a terminal in $N(B_1)$ and a terminal in $N(B_2)$. Then we must have $|s_1s| \leq B$ and $|s_2s| \leq B$.

**Upper bounds.** The new branch tree $B$ interconnects the root $s$ with the terminals $N(B_1)$ and $N(B_2)$. Since $B$ is assumed to be part of some minimum fixed orientation Steiner tree, it must have minimum length. A number of different heuristics can be applied to provide an upper bound on the length of a Steiner tree that interconnects $\{s\} \cup N(B_1) \cup N(B_2)$. For example, the following is a simple upper bound:

$$U(B_1, B_2) = |B_1| + \min_{t_1 \in N(B_1)} \|t_1 s\| + |B_2| + \min_{t_2 \in N(B_2)} \|t_2 s\|.$$

The branch tree $B$ can now be pruned if $U(B_1, B_2) < |B|$. In GeoSteiner several upper bounds are computed — properly ordered so that the bounds that can be computed quickly are tried first [297]. As in the Euclidean case, the upper bound tests can be quite time consuming, so it is important to find the right balance between running time and pruning efficiency.

**Construction and generation of full Steiner trees**

Like a branch tree, an FST is also constructed by merging a pair of branch trees $B_1$ and $B_2$. The stems of $B_1$ and $B_2$ must intersect at an angle that forms a valid bent edge — or must simply overlap to create an FST with straight edges only. All of the above pruning tests for branch tree candidates can be applied in a similar fashion to test FST candidates; specifically, the tests based on direction sets, canonical forms, the lune property, bottleneck Steiner distance bounds and upper bounds all apply for the construction of an FST from two branch trees.

After applying these pruning tests, the resulting number of generated FSTs is almost linear in practice, and the size of the largest generated FST, in practice, is bounded by a constant as $n$ increases. Using this algorithm, Nielsen et al. [297] have solved randomly generated problem instances with up to 1000 terminals in less than one hour for $\lambda \leq 8$. On a modern computer, FST generation for 1000 terminals and $\lambda = 4$ takes around one minute.

## 2.7 Applications and extensions

Basically all known applications of the fixed orientation Steiner tree problem come from printed circuit design and chip design. (For a more detailed introduction to these two problems, particularly chip design, see Section 3.6 in Chapter 3.) Since the invention of integrated circuits, *Manhattan routing* has been the de facto standard in chip design. In Manhattan routing, wires run either in horizontal or vertical directions. On a given layer of the chip, almost all wires run in the same direction — the so-called *preferred direction* of the layer. Vias are used to connect wires across layers.

During the 1990s advances in manufacturing technologies made it possible to produce chips with more than two interconnect layers. The increased number of layers opened up the possibility of using alternative directions so as to improve the quality of the routing with respect to congestion, delay and power usage. As a general rule, decreasing total wire length improves all these quality measures, and may even lead to a reduced chip size.

In this section we first discuss some early applications of non-Manhattan routing — namely printed circuit board routing and channel routing. Then we move on to discuss advantages and disadvantages of pervasive full-chip non-Manhattan routing. For further information about routing in chip design and extensions of the fixed orientation Steiner tree problem motivated by chip design, see Section 3.6 in Chapter 3.

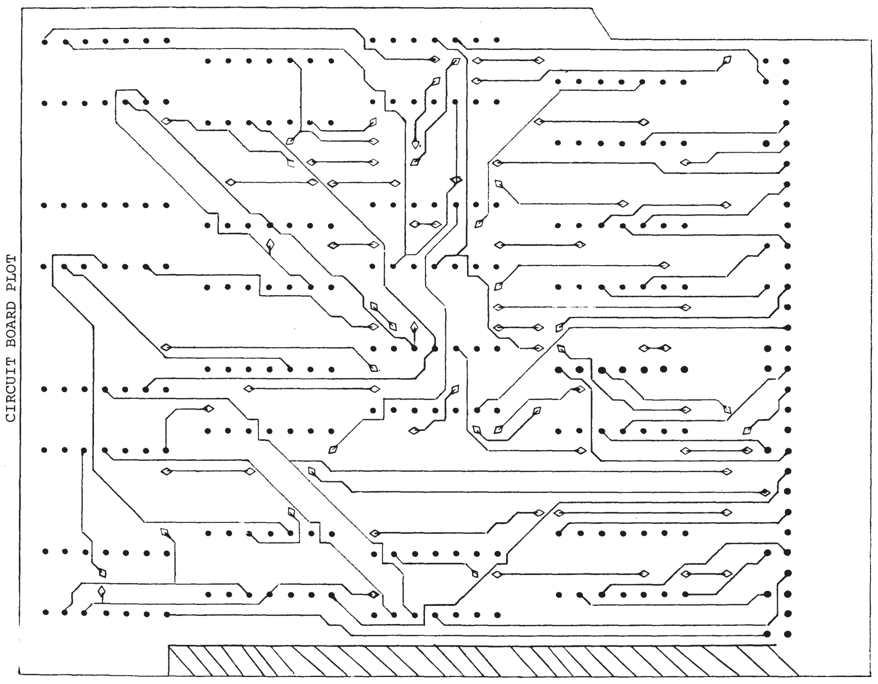


Figure 2.29: Printed circuit board with diagonal routing (from Heiss [193]).

**Printed circuit boards and channel routing**

The first algorithm for non-Manhattan routing in printed circuit design appears to be due to Heiss [193]. In 1968, he gave an extension of the classical Lee algorithm [244] that enabled *diagonal* routing (Figure 2.29). Heiss also gave a generalisation to more than two routing layers (where two layers correspond to the two surfaces of a double-sided printed circuit board). Another alternative to Manhattan routing was given by Chaudhuri [82] in 1979. Chaudhuri introduced routing with three uniform orientations (corresponding to $\lambda = 3$), and the new metric was denoted the 'Steiner metric' to distinguish it from the usual rectilinear (or Manhattan) metric. For printed circuit boards, Chaudhuri described a general routing scheme for two layers, and he presented a method to deal with the problem of routing three orientations on only two layers.

Chaudhuri also discussed non-Manhattan routing for the *channel routing problem*. Channel routing was one of the basic problems in chip design up to the late 1990s. In the technology of the time, cells were placed on rows on the chip surface, and routing was performed in the areas between these rows (the *channels*). A channel consists of two horizontal shores, where the terminals to be interconnected are located. Usually, each net consists of terminals from both shores; hence, the interconnection for each net has to cross the channel.

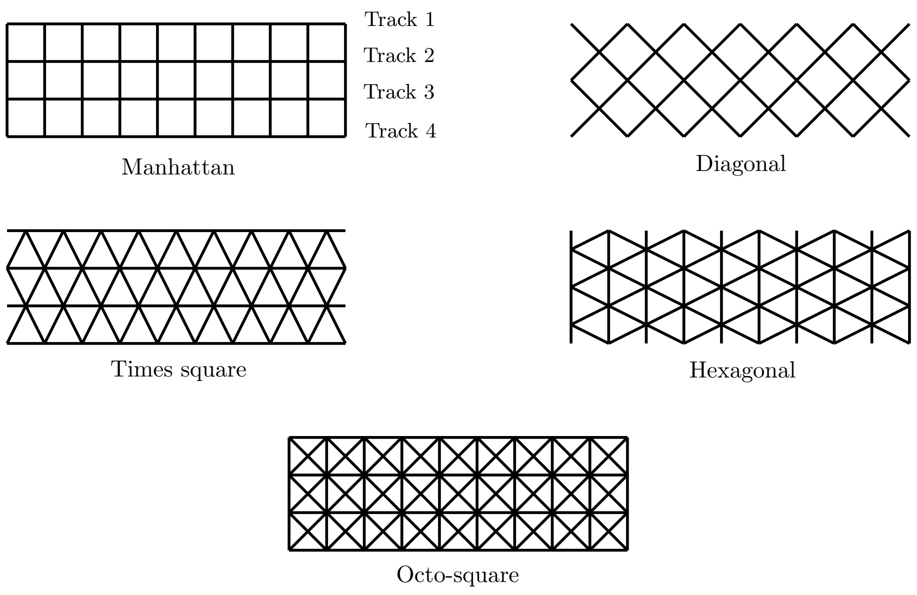


Figure 2.30: Channel routing models: Manhattan, diagonal, times square, hexagonal and octo-square.

In the traditional *Manhattan routing model*, routing is performed on a rectilinear grid with horizontal *tracks* and vertical *columns* (Figure 2.30). The number of tracks is called the *width* of the channel. The main objective of the channel routing problem is to minimise the number of tracks needed for the routing, as this minimises the area used by the channel. Since the objective is not to minimise the length of the nets (individually or jointly), channel routing may result in long connections for some of the nets. For a detailed introduction to the theory and algorithms for the Manhattan model, see [26, 251]. In the following we discuss some of the alternatives to the Manhattan model that have been considered in the literature.

In the *diagonal routing model*, the rectilinear grid is simply rotated $45°$ — hence there are still only two orientations [266, 267, 269] (Figure 2.30). One of the immediate advantages of this model is that short two-terminal interconnections (between two points on opposite horizontal shores) only require one layer change, as compared to the Manhattan model where two changes always are needed — except when the two terminals can be connected by a direct vertical connection. Thus fewer vias are in general needed for channel routing under this model.

Returning to the problem studied by Chaudhuri [82], where three uniform orientations are employed, one may distinguish between two cases. In the first case, the so-called *times square model*, one of the three orientations is the (usual) horizontal track, and the other two orientations are denoted right and left tracks, respectively [43, 268, 357, 363, 364]. In the second case, the *hexagonal model*, one of the three orientations is vertical [44, 315]. One of the advantages of the hexagonal model is that terminals can be spaced at the same interval that separates the wires in the grid; this is not possible under the times square model. Both models can be shown to have superior properties when compared to the Manhattan or diagonal models. Finally, the *octo-square model* merges the Manhattan and diagonal models. This model has four uniform orientations corresponding to $\lambda = 4$.

The octo-square model is discussed in a series of papers [26, 81, 83, 250, 294, 338, 384, 401, 421]. Although this model clearly has the advantage of having more available orientations, it also has the disadvantage that the rectilinear and diagonal wires cannot possibly have the same separation. A slightly different model was discussed by Chiang and Sarrafzadeh [95], who introduced $45^\circ$ wires locally to avoid so-called knock-knees in the wiring.

It is unclear to what extent the proposed models and algorithms have found their way into the design of real chips. Channel routing essentially became obsolete during the 1990s as a result of the new sea-of-cells technology, where cells could be placed (more or less) freely on the chip surface; also, over-the-cell routing became possible. In the next subsection we discuss general non-Manhattan routing which is relevant for current day microchip technology.

### General routing in chip design

The application of multiple orientations to the general routing problem in chip design was already anticipated by Widmayer et al. [402, 403] in 1985. In the early 1990s, Burman et al. [77] and Sarrafzadeh and Wong [341] gave the first practical applications of $\lambda$-geometry to general routing in chip design. During the following decade a series of heuristics — mostly with inspiration from rectilinear and Euclidean counterparts — were proposed for solving the Steiner tree problem in uniform and fixed orientation metrics.

In 2000 Koh and Madden [238] presented the first in-depth study of the feasibility of large-scale non-Manhattan routing architectures. Using simulation on realistic benchmarks they showed that average wire length reductions between $1\%$ and $11\%$ could be obtained for hexagonal routing ($\lambda = 3$) for a complete chip. Similarly, reductions between $6\%$ and $17\%$ could be obtained for octilinear routing ($\lambda = 4$). It should be noted that these improvements were obtained from the same placement of cells on the chip.

Choi et al. [97] presented a similar analysis for octilinear routing that confirmed the reductions in wire length; however, these reductions were obtained at the cost of an increase in the number of non-routed nets and an increase in the number of vias.

**X architecture** The increasing commercial interest in non-Manhattan routing during the 1990s led to the formation of the X Initiative in 2001, a consortium of software and chip companies that supported the development of the so-called *X architecture* [365]. The X architecture essentially adds diagonal wires to traditional Manhattan architecture (corresponding to $\lambda = 4$). Early test cases have shown that this architecture can lead to impressive reductions in area and path delay [216].

One of the major problems, however, with the X architecture is that gridded routing does not work in practice; rectilinear and diagonal wires do not have the same separation. This either results in problems with signal integrity or delay (if diagonal wires are too close or too thin), or results in suboptimal use of routing area (if only every second diagonal wire is used or the separation between rectilinear wires is increased). One solution to this problem — but an algorithmically challenging one — is to drop the preferred direction constraint and to allow all directions on all layers; this is denoted *liquid routing*. This allows for directional changes on a single layer, and can dramatically reduce the number of vias.

Ho et al. [200] suggested a multilevel approach for the X architecture. A multilevel algorithm consists of two main steps: coarsening followed by uncoarsening. The coarsening step is similar to the use of a global routing algorithm, but is iteratively employed. The algorithm of Ho et al. on average reduced wire length by 18.7% for a set of benchmark instances when compared to Manhattan routing.

**Y architecture** As a reaction to the shortcomings of the X architecture, Chen et al. [85, 86, 87, 88] in 2003 took one step back and investigated the use of hexagonal routing (corresponding to $\lambda = 3$); they coined this the *Y architecture*. The advantage is that gridded routing is in fact possible for this architecture since all parallel wires have the same separation. Therefore, from an algorithmic point of view, this architecture has a major advantage over the X architecture. Based on simulations under realistic scenarios, Chen et al. [87] estimated that the Y architecture improves wire length in the range $5-8\%$ over Manhattan architecture, while the X architecture obtains improvements in the range $9-11\%$. If the effect of decrease in routing area can be fully utilised to make the chip smaller, wire length improvements of approximately $23\%$ and $29\%$ are possible for the Y and X architecture, respectively.

A number of authors, including Yan [422] and Samanta et al. [336], have also proposed models for Steiner trees in X and Y architectures that attempt to optimise delay, rather than wire length.

Although it has been shown that both the X and Y architectures have the potential to decrease wire length and signal delay significantly, it is (as of this writing) unclear how many chips have been produced with these architectures. The first commercial chip (from Toshiba) using the X architecture was produced in 2004, and at least one chip (from ATI) followed in 2005.

**General architecture** In *general architecture* any number of uniform orientations can be used for routing. Since each routing layer has a preferred direction, the total number of available orientations usually depends on the number of layers. However, since more than 10 routing layers are already common, the number of available layers is not a limiting factor.

The problem of balancing the use of routing resources on the available layers was studied for the Manhattan architecture by Yildiz and Madden [426, 427], and for general architecture by Agnihotri and Madden [3]. The idea is to make the routing cost on each layer (as seen by the routing algorithm) depend on the congestion on the layer. By iteratively adjusting the routing cost on each layer, congestion can be lowered on highly utilised layers.

Paluszewski et al. [302, 303] presented a completely different, geometry-oriented approach to deal with congestion when many routing layers are available. The idea is to exploit the fact that a minimum Steiner tree can be embedded in many different ways in $\lambda$-geometry. In the first phase of the algorithm, minimum Steiner trees and their *flexibility polygons* are computed for each net on the chip; recall that a flexibility polygon is a geometric representation of all minimum Steiner trees for a given net (see Section 2.4.3). Each flexibility polygon is assigned a weight that is equal to the routing area used by the minimum Steiner tree divided by the area of the flexibility polygon. The weight represents the average probability that routing resources are needed for a given point in the flexibility polygon. Based on the weights of the flexibility polygons, a *congestion map* is constructed for the whole chip area. The congestion map gives the estimated routing resources needed for each point of the chip area. The idea of the algorithm is now to move wires away from highly congested areas. Experiments with the new method show that it is indeed an advantage to use flexibility polygons in the initial routing phase. When using 5 or 6 layers, architectures with $\lambda \geq 4$ reduce total wire length by$7 - 18\%$ when compared to Manhattan routing.

## Exercises

**2.1.** If $\mathcal{P}_1 = u_1u_2u_3u_4$ is a convex fixed orientation path between $u_1$ and $u_4$, then it is non-minimal.

**2.2.** Show that if we choose a set of legal orientations such that the orientation polygon $\mathcal{C}$ is not convex, then for any pair of points $p$ and $q$ and for any orientation corresponding to a vertex of $\mathcal{C}$ that is not an extreme point of the convex hull of $\mathcal{C}$ there exists a shortest fixed orientation path between $p$ and $q$ not using that orientation.

**2.3.** Prove Corollary 2.4.

**2.4.** Show that in any minimum $\lambda$-geometry Steiner tree the minimum meeting angle at any Steiner point or terminal is strictly greater than $\pi/2 - 2\omega$ (without using Theorem 2.5).

**2.5.** Show that, for any given $\lambda$, there exists a minimum $\lambda$-geometry Steiner tree such that the minimum meeting angle $\phi$ at a Steiner point satisfies $\phi < 2\pi/3$. [Hint: For the case where $\lambda = 3m$ this requires a continuity argument. Consider a minimum $\lambda$-geometry Steiner tree with a single Steiner point where all angles at the Steiner point are $2\pi/3$, and investigate the effect of perturbing one of the terminals at right angles to the incident edge.]

**2.6.** Show that, despite Corollary 2.6, if $\lambda = 3$ or $6$, then for any terminal set:

(a) no minimum $\lambda$-geometry Steiner tree has a Steiner point of degree $5$ or $6$; and

(b) there exists a minimum $\lambda$-geometry Steiner tree in which every Steiner point has degree $3$.

**2.7.** Show that if a direction set $\mathcal{D}$ has cardinality $4$, then the angle $\theta$ between the green and blue directions is strictly less than $\pi$ (Lemma 2.13). [Hint: Apply a continuity argument similar to that used in the proof of Lemma 2.7.]

**2.8.** Let $s$ be a Steiner point in a fixed orientation Steiner configuration with coloured direction set $\mathcal{D} = \{\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_k\}$, where $k = 5$ or $6$. Define the *direction weight set* of $\mathcal{D}$ to be the set $\{w_1, w_2, \ldots, w_k\}$ where each $w_i = 1/|\mathbf{d}_i|$, and define the *direction angle set* of $\mathcal{D}$ to be the set $\{\theta_1, \theta_2, \ldots, \theta_k\}$ where each $\theta_i$ is the angle between $\mathbf{d}_i$ and $\mathbf{d}_{i+1}$ (where addition in the subscripts is modulo $k$).

(a) If $k = 5$ show that

$$\frac{w_2\sin(\theta_5) - w_1\sin(\theta_1+\theta_5)}{\sin(\theta_1)} + \frac{w_3\sin(\theta_4) - w_4\sin(\theta_3+\theta_4)}{\sin(\theta_3)} = w_5.$$

(b) If $k = 6$ show that the following two conditions hold simultaneously:

$$\frac{w_2\sin(\theta_5+\theta_6) - w_1\sin(\theta_1+\theta_5+\theta_6)}{\sin(\theta_1)} + \frac{w_3\sin(\theta_4) - w_4\sin(\theta_3+\theta_4)}{\sin(\theta_3)} = w_5$$

and

$$\frac{w_2\sin(\theta_6) - w_1\sin(\theta_1+\theta_6)}{\sin(\theta_1)} + \frac{w_3\sin(\theta_4+\theta_5) - w_4\sin(\theta_3+\theta_4+\theta_5)}{\sin(\theta_3)} = w_6.$$

**2.9.** Prove Corollary 2.18.

**2.10.** Let $T$ be a fulsome $\lambda$-geometry Steiner tree, where $\lambda = 3m$. Show that any fundamental 1-point zero-shift for $T$ preserves the amount of primary material in $T$.

**2.11.** Given a fixed orientation metric with unit circle $\mathcal{C}$, let $\mathbf{u}_l$, $l = 0, \ldots, 2\sigma - 1$, be the $2\sigma$ vectors that define the extreme points $\mathcal{C}$ (in counter-clockwise order around the circle). For two given points $p$ and $q$, let $\{\alpha_l, \beta_l\}$ be the unique solution to $q = p + \alpha_l \mathbf{u}_l + \beta_l \mathbf{u}_{l+1}$ for each $l = 0, \ldots, 2\sigma - 1$. Show that $\|pq\| = \max_{l \in \{0,\ldots,2\sigma-1\}}(\alpha_l + \beta_l)$.

**2.12.** Complete the proof of Lemma 2.25, by showing that if $(s, u)$ and $(s', v)$ use different red directions, then $(s, v)$ and $(s', u)$ intersect. [Hint: Show that there exist real numbers $p, q > 0$ such that: $\alpha_1 = 1+p,\ \ \alpha_2 = -q,\ \ \beta_1 = -p,\ \ \beta_2 = 1+q$, and derive suitable expressions for $k_1$ and $k_2$ in terms of $p$ and $q$ that satisfy Equation (2.3).]

**2.13.** Show that $\cos(\pi/(2\lambda))^{-1}$ is the maximum ratio of the distance between two points in $\lambda$-geometry and Euclidean geometry.

**2.14.** Show that, in the process of generating an FST for a minimum fixed orientation Steiner tree, when merging two given branch trees to create a new branch tree there are at most three possible directions for the stem of the merged branch tree.

**2.15.** Note that the triangular feasible region for the Steiner point in Figure 2.19 includes one of the terminals of the tree. This implies that the minimum Steiner tree shown is not fulsome. Show that, for $\lambda = 3$ and any set of three terminals, any non-degenerate minimum Steiner tree that contains a Steiner point is not fulsome.

## Notes

[25]The results of Widmayer et al. [402, 403] were also anticipated, in part, by earlier and contemporary papers such as [410] and [387].

[26]The norms associated with the class of fixed orientation metrics have been referred to in the literature as ‘polyhedral norms’, in [410], or ‘block norms’, in [387]. Moreover, if we relax the condition that the polygon $\mathcal{C}$ is centrally symmetric, then rather than having an associated norm, we have an associated *gauge*; see [146] for more details.

[27]The first proof of Theorem 2.5 was given by Sarrafzadeh and Wong [341], albeit not covering the case where $\lambda$ is a multiple of 3 correctly. Alternative (and correct) proofs using various proof techniques were given by Koh [237] (only for $\lambda = 4$), Li et al. [258] (only lower bounds), Brazil et al. [65], Swanepoel [361], Hayase [191] and Il’yutko [219].

[28]Note that in $\lambda$-geometry these direction sets are the same as the ‘feasible direction sets’ defined in [70].

[29]This theorem can be strengthened to include a lower bound of $\sigma$ on the number of complementary coloured direction sets. This follows by proving a converse to Theorem 1.28 showing that every set of supporting lines satisfying the centroid property corresponds to a Steiner configuration. The details can be found in [74], where some of the arguments rely on results from [277]. The main consequence of this lower bound is to show that the time complexity of the linear-time enumeration algorithm developed at the end of this section cannot be improved.

[30]Zero-shifts were introduced by Du and Hwang [137] for $\lambda = 3$, and originally used as a technical tool in the quest for better bounds on the size of the generalised Hanan grid [237, 246, 247].

[31]In 1999, Thurber and Xue [371] gave a linear programming formulation for the $\lambda = 3$ case, and in 2002, Xue and Thulasiraman [419] generalised the formulation to the general uniform orientation metric. Zachariasen [432] pointed out a non-trivial error in this formulation and presented a new and correct formulation, now generalised to all fixed orientation metrics. This latter formulation is the one briefly presented here.

[32]Subsequently Li et al. [257] gave a simple algorithm to construct this triangular region based on finding median points and so-called mid-orientation lines. More generally, Shen [349] and Hayase [191] independently showed that when $\lambda$ is a multiple of 3, the feasible region (called a 'public domain' in [349] and a 'diamond area' in [191]) is a convex polygon with up to six vertices; when $\lambda$ is not a multiple of 3, then the minimum Steiner tree for three terminals is unique and the feasible region contains a single point.

[33]This theorem has been shown over time to hold for increasingly larger classes of fixed orientation problems. In 1992 Du and Hwang [137] proved that Theorem 2.29 holds for $\lambda = 3$ (uniform metric with three orientations). They also conjectured that for any $i > 0$ there exists a fixed orientation metric and a terminal set $N$ such that all minimum Steiner trees for $N$ have some Steiner point *not* in $\mathbf{GG}_i(N)$. In other words, they conjectured that the bound $n-2$ in Theorem 2.29 cannot be reduced to a constant. In 1995 Koh [237] and Lee et al. [247] independently proved that the theorem holds for $\lambda = 4$ by showing that it is always possible to perform zero-shifts such that some Steiner point is connected to two terminals using straight edges only. In 1996 Lee and Shen [246] generalised the result to any $\lambda$ (or uniform orientation metric) using the same proof technique. In 2001 Li et al. [259] showed that Theorem 2.29 holds for all unweighted fixed orientation metrics. Finally, in 2009 Brazil and Zachariasen [74] implicitly proved that the theorem also holds for the weighted case; this was first stated explicitly later that year by Zachariasen [433].

# Chapter 3

# Rectilinear Steiner Trees

In this chapter we consider the problem of constructing a network of minimum length interconnecting a given set of points in the Euclidean plane, where each edge of the network is composed of *horizontal* and *vertical* line segments. This problem is known as the *rectilinear* Steiner tree problem. Of all the problems studied in this book, the rectilinear Steiner tree problem is probably the most important from an applications point of view. Since its introduction by Maurice Hanan [189] in 1966, the problem has been recognised as having a crucial role in chip design, in particular in the physical design of very-large-scale integration (VLSI) circuits. In chip design the given points correspond to electrical terminals that should be interconnected using a minimum amount of wire. An overview of current applications is given at the end of this chapter.

For any two points $p$ and $q$ in the plane, the minimum length of a path between them composed of horizontal and vertical line segments defines a norm, known as the $\ell_1$ *norm* (or $\ell_1$ *distance*). If $p = (p_x, p_y)$ and $q = (q_x, q_y)$ in the plane, their $\ell_1$ *distance* is $|pq|_1 = |p_x - q_x| + |p_y - q_y|$, that is, the sum of distances in each of the two dimensions. The $\ell_1$ distance is also called the *rectilinear* or *Manhattan* or *taxicab* distance. Formally, the rectilinear Steiner tree problem is as follows:

RECTILINEAR STEINER TREE PROBLEM IN THE PLANE
**Given**: A set of points $N = \{t_1, \ldots, t_n\}$ lying in the plane.
**Find**: A geometric network $T = (V(T), E(T))$, such that $N \subseteq V(T)$, and such that $|T|_1 := \sum_{e \in E(T)} |e|_1$ is minimised.

A solution to this problem is always a tree, and is referred to as a *minimum rectilinear Steiner tree*. The given points in $N$ are denoted *terminals*, and the possible points in $V(T) \setminus N$ are called *Steiner points*. The rectilinear Steiner tree problem is clearly identical to the uniform orientation Steiner tree problem for $\lambda = 2$ (see Chapter 2). In this chapter we always consider a minimum rectilinear Steiner tree as being embedded in the Euclidean plane using horizontal and vertical line segments. This means that all lengths can be measured using the Euclidean metric.

Note that although the theory of rectilinear Steiner trees builds in a natural way on many of the results in the previous two chapters, we will nevertheless present this material in a way that is as self-contained as possible.

## 3.1 Local properties of Steiner points and full components

We begin our study with some definitions that are useful for characterising the structure of minimum rectilinear Steiner trees, and we recall a number of the relevant definitions that were introduced in Chapters 1 and 2. We then establish some basic geometric properties of minimum rectilinear Steiner trees, and we end the section with a powerful characterisation of the full components of a minimum rectilinear Steiner tree — the so-called Hwang form.

### 3.1.1 Basic definitions and properties

#### Basic definitions

Consider a minimum rectilinear Steiner tree $T = (V(T), E(T))$ for a given terminal set $N$; an example of such a tree is given in Figure 3.1. The node set $V(T)$ contains all elements of $N$ and some additional Steiner points. We can assume without loss of generality that all Steiner points have degree 3 or 4. A Steiner point of degree 3 is called a *T-point*, and a Steiner point of degree 4 is called a *cross*.[34]

The edge set $E(T)$ consists of edges that connect pairs of nodes $u$ and $v$ by shortest rectilinear paths. The edge $(u, v)$ is a *straight edge* if $uv$ is either a horizontal or a vertical line segment; otherwise, $(u, v)$ is a *bent edge*. As shown by Theorem 2.1 in Chapter 2, a bent edge can be assumed to consist of exactly two line segments. Therefore, we may assume that a rectilinear bent edge consists of a horizontal and a vertical line segment that meet at a *corner point*. For any bent edge $(u, v)$ there are two possible minimum length embeddings that contain a single corner point; this is illustrated in Figure 3.2. We describe the process of moving from one of these embeddings to the other as a *flip*.

A *line of segments* is a sequence of one or more adjacent, collinear segments with no terminal nodes sharing two adjacent segments (however, the endpoints of the line may be terminals). This leads to two definitions that are important for developing canonical forms for minimum rectilinear Steiner trees.

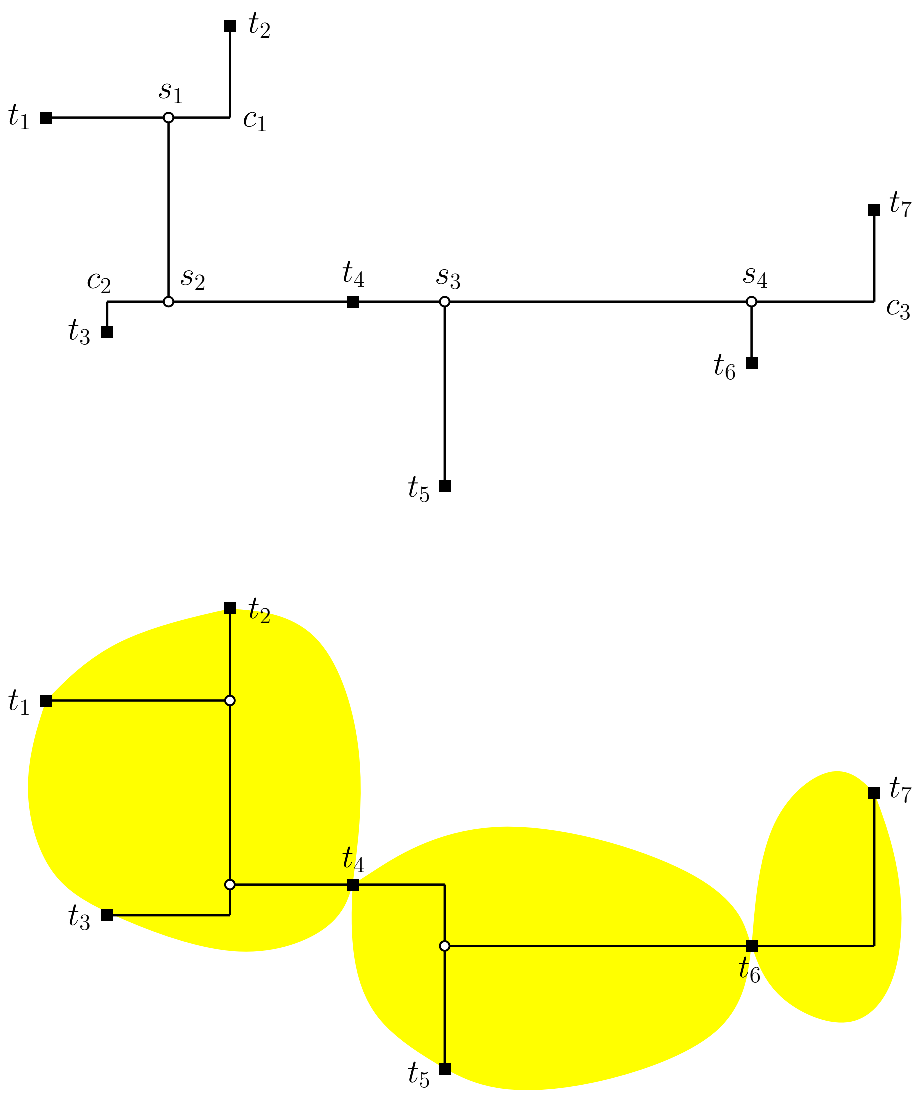


Figure 3.1: Two minimum rectilinear Steiner trees for the same set of terminals $t_1, \ldots, t_7$. In the top tree, $s_1$, $s_2$, $s_3$ and $s_4$ are Steiner points, while $c_1$, $c_2$ and $c_3$ are corner points. Edge $(s_1, s_2)$ is a straight edge and edge $(s_4, t_7)$ is a bent edge; line $c_3t_4$ is a complete line and $(c_3t_4, c_3t_7)$ is a complete corner. The top tree has two full components and is not fulsome. The bottom tree has three full components and is fulsome.

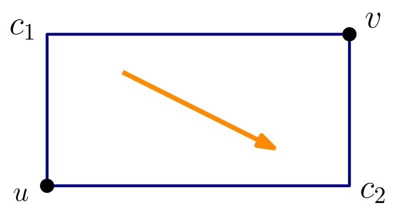


Figure 3.2: Two possible embeddings of an edge $(u, v)$, one containing corner point $c_1$ and the other containing corner point $c_2$. The orange arrow indicates a flip between one embedding and the other.

**Definitions [Complete line, complete corner]**: A *complete line* is a line of segments of maximal length; it is not properly contained in any other line of segments. Any corner point $c$ is an endpoint of two complete lines, one in each of the two perpendicular directions given by the incident segments. Let $t$ and $t'$ be the other endpoints of the complete lines incident to $c$. The pair of complete lines ($ct$, $ct'$) is called a *complete corner* located at $c$; $ct$ and $ct'$ are the *legs* of the complete corner.

An example of a complete corner is $(c_3t_4, c_3t_7)$ in Figure 3.1 (top).

As in the previous chapters, a rectilinear Steiner tree in which every terminal has degree 1 is called a *full* rectilinear Steiner tree. Every rectilinear Steiner tree is a union of full rectilinear Steiner trees (also known as *full components*) meeting only at terminals. A rectilinear Steiner tree is said to be *fulsome* if it has the maximum possible number of full components amongst all rectilinear Steiner trees with the same length. (Hence, a minimum rectilinear Steiner tree is full and fulsome if there is no minimum rectilinear Steiner tree on the same set of terminals with two or more full components.)

## Properties of Steiner points

The following lemma summarises some basic properties of Steiner points in minimum rectilinear Steiner trees; the proof is left as Exercise 3.1.

**Lemma 3.1** *Let $s$ be a Steiner point in a minimum rectilinear Steiner tree $T$. Then the following properties are true:*

- *The edges incident to $s$ cannot overlap with each other for any embedding of the edges.*
- *If $s$ is a cross, then all edges incident to $s$ are straight edges.*
- *If $s$ is a T-point, then at most one edge incident to $s$ is a bent edge.*

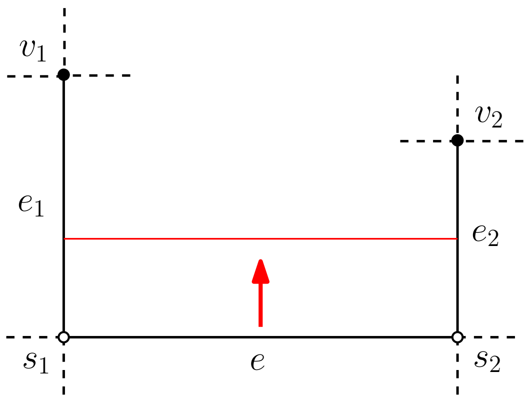


Figure 3.3: Illustration of rectilinear sliding lemma.

- *For any straight edge $(s, u)$ incident to $s$ there exists another straight edge $(s, v)$ incident to $s$ and perpendicular to $(s, u)$.*

In order to achieve a useful geometric characterisation of the Steiner points in a minimum rectilinear Steiner tree $T$, we assume in the remainder of this section that $T$ is *fulsome*. We begin with a technical lemma that is a direct corollary of Lemma 1.37 in Chapter 1; for completeness, we sketch the proof of the lemma here.

**Lemma 3.2** ***[Rectilinear sliding lemma]*** *Let $e = (s_1, s_2)$ be a straight edge connecting two Steiner points $s_1$ and $s_2$ in a fulsome minimum rectilinear Steiner tree $T$. Let $e_1 = (s_1, v_1)$ be the next edge incident to $s_1$ travelling counter-clockwise from $e$, and let $e_2 = (s_2, v_2)$ be the next edge incident to $s_2$ travelling clockwise from $e$. Suppose $e_1$ and $e_2$ are straight edges, perpendicular to $e$, and located on the same side of the line through $e$ (Figure 3.3). Then $T$ is not fulsome.*

**Proof.** (Sketch) We define a *slide* of a straight edge in a rectilinear Steiner tree to be a displacement of that edge in a direction orthogonal to the direction of the edge. Assume that we slide $e$ so that its endpoints move in the direction of $v_1$ and $v_2$ (see Figure 3.3). As we slide $e$, the endpoints will overlap with the nodes on the perpendicular complete lines containing $s_1$ and $s_2$, respectively. If $e$ meets a terminal during the slide, then $T$ is not fulsome. Therefore, $v_1$ and $v_2$ must be Steiner points. None of the Steiner points met by $e$ can have an edge that overlaps with $e$; thus, it must be possible to continue the slide past any Steiner point. Also, it is easy to show that none of the subsequent nodes met by $e$ (after $v_1$ and $v_2$) can be corner points, as this would contradict the minimality of $T$. Hence, we can continue the slide of $e$ until we obtain a contradiction to either minimality or fulsomeness. ∎

This lemma is called the rectilinear sliding lemma, since it can be restated as follows: in a fulsome minimum rectilinear Steiner tree we cannot slide any straight edge freely between two perpendicular straight edges, as this leads to a contradiction to either minimality or fulsomeness. (Note that such a slide is an example of a zero-shift that does not change the length of the tree; see Section 2.3.3.)

Together, Lemmas 3.1 and 3.2 can be used to obtain the following properties of Steiner points in fulsome minimum rectilinear Steiner trees:

**Lemma 3.3** ***[A cross has only terminals as neighbours]*** *Let $s$ be a cross in a fulsome minimum rectilinear Steiner tree $T$. Then the neighbours of $s$ are terminals.*

**Proof.** The four edges incident to $s$ are straight edges (Lemma 3.1). Assume that one of the neighbours of $s$, denoted by $u$, is a Steiner point. By Lemma 3.1, Steiner point $u$ must have an incident straight edge $(u, x)$ that is perpendicular to $(s, u)$. Now, edge $(s, u)$ fulfils the conditions of the rectilinear sliding lemma, contradicting the fulsomeness of $T$. ∎

**Lemma 3.4** ***[A T-point ends in a terminal]*** *Let $s$ be a T-point in a fulsome minimum rectilinear Steiner tree $T$, and let $u$, $v$ and $w$ be the three neighbouring nodes of $s$. Suppose the edges $(s, v)$ and $(s, w)$ are collinear and straight edges. Then $(s, u)$ is a straight edge and $u$ is a terminal.*

**Proof.** First we observe that $(s, u)$ must necessarily be a straight edge — otherwise it is clear, by applying a flip to $(s, u)$, that $T$ is not length-minimal. Assume that $u$ is a Steiner point. Now we can use the same arguments as in the proof of Lemma 3.3 to show that $T$ is not fulsome. ∎

In other words, if a degree 3 Steiner point has two collinear incident edges, then the third edge must be connected to a terminal. Note that the lemma does not hold if one of the neighbours $v$ or $w$ is a corner point.

The proof of the third corollary of Lemmas 3.1 and 3.2, below, is left as an exercise (Exercise 3.2).

**Lemma 3.5** ***[A complete corner ends in terminals]*** *Let $c$ be a corner point for a bent edge in a fulsome minimum rectilinear Steiner tree $T$, and let ($ct$, $ct'$) be the complete corner located at $c$. Then $t$ and $t'$ are both terminals of $T$.*

### 3.1.2 Hwang form for full components

The rectilinear distance function induces a metric, known as the *rectilinear metric*. The rectilinear metric is not strictly convex, so in general there may be infinitely many minimum rectilinear Steiner trees for a given set of terminals. It is therefore important from an

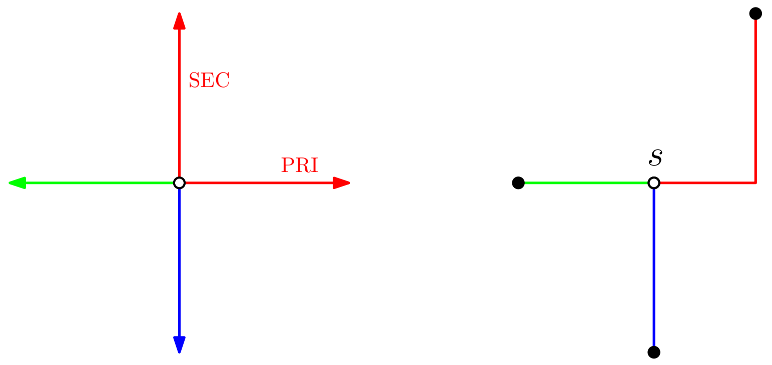


Figure 3.4: Rectilinear direction set (left) and a Steiner point $s$ using the direction set (right).

algorithmic point of view to devise canonical forms of minimum rectilinear Steiner trees that can be constructed efficiently. In this section we focus on the local structure of full components, and we show that full and fulsome minimum rectilinear Steiner trees can be assumed to have a simple canonical form called the *Hwang form*.[35]

As a consequence of Lemma 3.3, we first focus on full components where all Steiner points have degree 3 (i.e., are T-points). Later we shall see how crosses — which are special full components that have a single Steiner point of degree 4 — fit into the picture.

The approach that we follow in this section is as follows. First we recall and apply the relevant parts of the general theory developed in Chapter 2 for fixed orientation metrics to the rectilinear metric. Then we show that the general theory leads to a particularly simple characterisation of full components for the rectilinear metric, immediately implying the Hwang form property.

### Direction sets, primary/secondary directions, zero-shifts and one bent edge

Recall that the legal directions that are used by the edges incident to a Steiner point are part of some *direction set* (where directions are considered as oriented either inward or outward from the Steiner point). For the rectilinear metric, a Steiner point of degree 3 has at most one incident bent edge (Lemma 3.1). Therefore, a direction set has four directions: two *red* directions corresponding to the (possibly) bent edge, one *green* and one *blue* direction (Figure 3.4). The two red directions are labelled the exclusively primary and exclusively secondary direction, respectively, in counter-clockwise order around the Steiner point; the blue and green edges can be considered to be both primary and secondary. Note that all the figures in this section are illustrated using the specific direction set shown in Figure 3.4.

We can use *zero-shifts* (as introduced in Section 2.3.3) to make length-preserving perturbations of the Steiner points. In the rectilinear metric, zero-shifts are obtained by performing a series of slides of straight edges (as in Figure 3.3) and/or flips of bent edges. Zero-shifts can be decomposed into perturbations that exchange exclusively primary and exclusuvely secondary material in a pair of edges. A zero-shift is *complete* if it uses all of the exclusively primary or all of the exclusively secondary material from one of the pair of edges. (Hence, a zero-shift has the effect of making at least one bent edge straight.)

Let $T$ be a full and fulsome minimum rectilinear Steiner tree. From Chapter 2 we have the following results:

1. $T$ uses a single direction set (Theorem 2.11).
2. Given any two red edges $e_1$ and $e_2$ in $T$, where $e_1$ has an exclusively primary component and $e_2$ has an exclusively secondary component, there exists a complete zero-shift for $e_1$ and $e_2$ (Theorem 2.19).
3. There exists a minimum rectilinear Steiner tree with the same terminals and topology as $T$ that has at most one bent edge (Corollary 2.20).

The fact that $T$ may be assumed to have at most one bent edge can be used to prove a powerful characterisation of $T$. We make use of the following definition.

**Definition [Caterpillar tree]**: Define a *caterpillar tree* to be a tree that has a central path $\mathcal{P}$ such that every node in the tree is either on $\mathcal{P}$ or is connected directly to $\mathcal{P}$.

In other words, a full Steiner tree $T$ is a caterpillar tree if and only if the subtree induced by the Steiner points of $T$ is a path. Note that being a caterpillar is a property of the topology of the Steiner tree.

**Lemma 3.6** ***[Rectilinear full components are caterpillar trees]*** *Let $T$ be a full and fulsome minimum rectilinear Steiner tree spanning at least 3 terminals, and containing at most one bent edge. Then the topology of $T$ is a caterpillar tree where the central path is formed by all the Steiner points in $T$.*

**Proof.** We begin by showing that every Steiner point in $T$ has at least one terminal as neighbour (or, equivalently, at most two adjacent Steiner points). First consider a Steiner point $s$ that is incident to straight edges only. By Lemma 3.4, Steiner point $s$ has at least one adjacent terminal.

Next, if $T$ contains a bent edge, consider a Steiner point $s$ that is incident to the single bent edge in $T$ (Figure 3.5). Let $(s, u)$ and $(s, v)$ be the two other (straight) edges that are incident to $s$. We show that either $u$ or $v$ is a terminal. Both $u$ and $v$ are incident

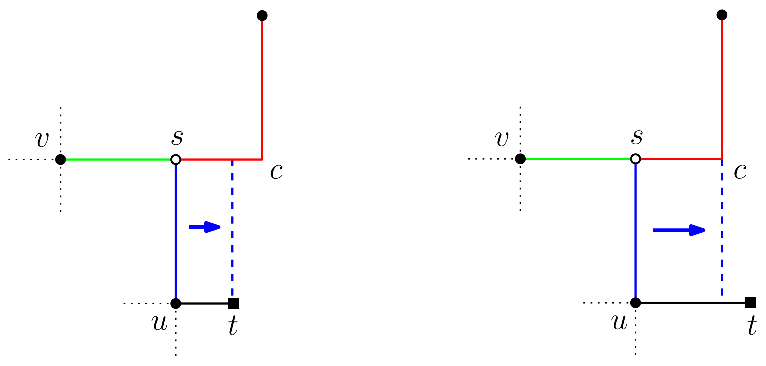


Figure 3.5: Illustration of Lemma 3.6. On the left $t$ is closer to $su$ than $c$. On the right $t$ is further away from $su$ than $c$.

to straight edges only, since $T$ has at most one bent edge. We can assume that the bent edge is embedded so that its corner point $c$ is collinear with $s$ and $v$, as in Figure 3.5.

If $u$ is a Steiner point then, by Lemma 3.4, $u$ must have an adjacent terminal $t$, such that $(u, t)$ is perpendicular to $(s, u)$. If $t$ is on the same side of $su$ as $v$, then the rectilinear sliding lemma is violated. Instead, assume that $t$ is on the side of $su$ opposite to $v$, as in Figure 3.5. If $t$ is closer to $su$ than the corner point $c$ of the bent edge, then we may slide $su$ towards $t$: the slide can be continued until the edge meets $t$ — which contradicts the fulsomeness of $T$ (Figure 3.5, left). If $t$ is further away from $su$ than $c$, then the slide can be continued until an endpoint of the edge meets $c$ (Figure 3.5, right). In this case $c$ becomes a T-point where the neighbour $v$ must be a terminal (by Lemma 3.4). Thus, we have shown that either $u$ or $v$ must be a terminal.

Let $\mathcal{T}$ be the topology of $T$. Note that, because $\mathcal{T}$ is full, the subgraph of $\mathcal{T}$ induced by the Steiner points is a tree (in other words, is connected). Since each Steiner point is adjacent to at most two other Steiner points, this tree must be a path, completing the proof. ∎

We note that the condition in Lemma 3.6 that $T$ has only one bent edge can be omitted, as it can be shown that if $T$ is a full and fulsome minimum rectilinear Steiner tree, then any slide (and hence any zero-shift) in $T$ preserves the topology of $T$. See Exercise 3.3.

Recall from Chapter 2 that a subtree of $T$ is a primary subtree (or secondary subtree), if all edges are primary (respectively, secondary) edges. A subtree that is either primary or secondary is denoted a *clean* subtree.

**Lemma 3.7** ***[Subtrees consisting of straight edges only are clean]*** *Let $T$ be a full and fulsome minimum rectilinear Steiner tree. Consider any subtree $T'$ of $T$ that consists of straight edges only. Then $T'$ is a clean subtree.*

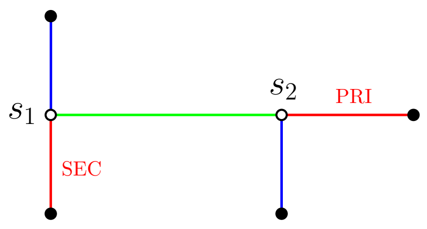


Figure 3.6: Illustration of proof of Lemma 3.7. Exactly one of the two red edges is collinear with $(s_1, s_2)$.

**Proof.** Assume that $T'$ is not clean; that is, there exist both a primary red edge and a secondary red edge in $T'$. Then there must exist a pair of neighbouring Steiner points $s_1$ and $s_2$ in $T'$, such that $s_1$ has an incident primary red edge and $s_2$ has an incident secondary red edge. It follows that exactly one of these red edges is collinear with $(s_1, s_2)$; hence, either $s_1$ or $s_2$ is a T-point where the non-collinear incident edge does not end in a terminal (Figure 3.6). By Lemma 3.4, this gives a contradiction. ∎

## Canonical forms

Consider any ordering $\phi$ of the edges of a full and fulsome minimum rectilinear Steiner tree $T$. From Theorem 2.23 in Chapter 2 we know that there exists a minimum rectilinear Steiner tree $T_\phi$ with the same terminals and topology as $T$ with the following properties: $T_\phi$ contains an edge $e_\phi$, which we refer to as a *transition edge*, satisfying the following properties:

- all edges other than the transition edge are straight edges;
- all edges that come before the transition edge under the given ordering are primary;
- all edges that come after the transition edge under the given ordering are secondary.

We say that $T_\phi$ is *canonical* with respect to the given ordering $\phi$ of the edges.

Consider a full and fulsome minimum rectilinear Steiner tree $T$ that is canonical under some ordering of the edges, and let $e$ be the transition edge. Let $T_1$ and $T_2$ be the two connected components of $T - e$ (the forest obtained from $T$ by deleting the edge $e$). Note that these two subtrees consist of straight edges only. From Lemma 3.7 it follows that $T_1$ and $T_2$ are clean subtrees, and therefore each has one of the two forms that are illustrated in Figure 3.7. Hence, each subtree consists of a complete line containing the Steiner points with *alternating segments* attached to the complete line.

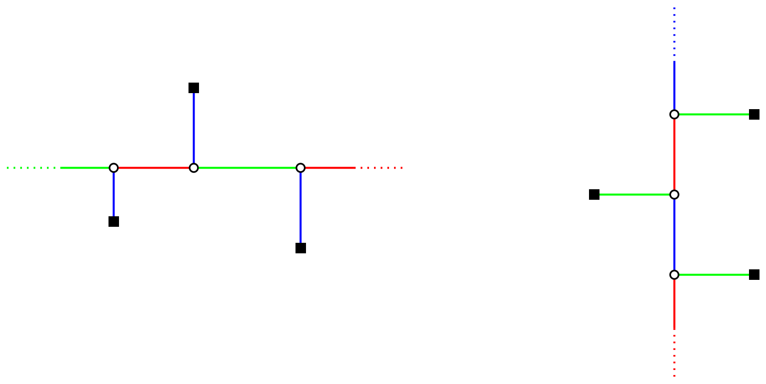


Figure 3.7: A primary and a secondary clean subtree for the direction set shown in Figure 3.4.

### The Hwang form

So far we have shown that for a full and fulsome minimum rectilinear Steiner tree $T$ that is canonical under some ordering of the edges, the two subtrees $T_1$ and $T_2$, obtained by deleting the transition edge $e$, are clean subtrees. In order to achieve a particularly simple canonical form, we now restrict the given ordering of the edges in the canonical form as follows. Consider the Steiner point path $\mathcal{P}$ of $T$, and let $t$ be a terminal that is connected directly to one of the *endpoints* of $\mathcal{P}$. We order the edges by a depth-first traversal of $T$ starting at $t$. When we visit a Steiner point during the traversal we always visit the adjacent terminal leaf or leaves before visiting the single Steiner point child (if any). Informally, we visit the nodes of $T$ in the order in which they appear along the central path $\mathcal{P}$ of $T$. Denote this ordering of the edges by $\phi = \phi(t)$, and let $e_\phi$ be the transition edge of the tree $T_\phi$ having the resulting canonical form.

First, assume that $e_\phi$ is a bent edge, and let subtrees $T_1$ and $T_2$ be the two connected components of $T - e_\phi$, where $T_1$ contains terminal $t$. Observe that $e_\phi$ must either be an edge on the central path $\mathcal{P}$ or incident to one of the endpoints of $\mathcal{P}$; this follows from the arguments in the proof of Lemma 3.6. Therefore, from Lemma 3.7 we have that $T_1$ is a primary (clean) subtree and $T_2$ a secondary (clean) subtree.

**Claim.** One of the subtrees $T_1$ and $T_2$ spans at most 2 terminals.

**Proof of Claim.** Assume, to the contrary, that both $T_1$ and $T_2$ span more than 2 terminals (as illustrated in Figure 3.8, left). This implies that $T_1$ contains an exclusively primary red edge and $T_2$ contains an exclusively secondary red edge. Then we can always perform a simple zero-shift such that we obtain a new minimum rectilinear Steiner tree $T'$, where another edge $e'$ is the only bent edge (Figure 3.8, right). Furthermore, in $T'$ one of the subtrees is no longer clean, contradicting Lemma 3.7. ∎

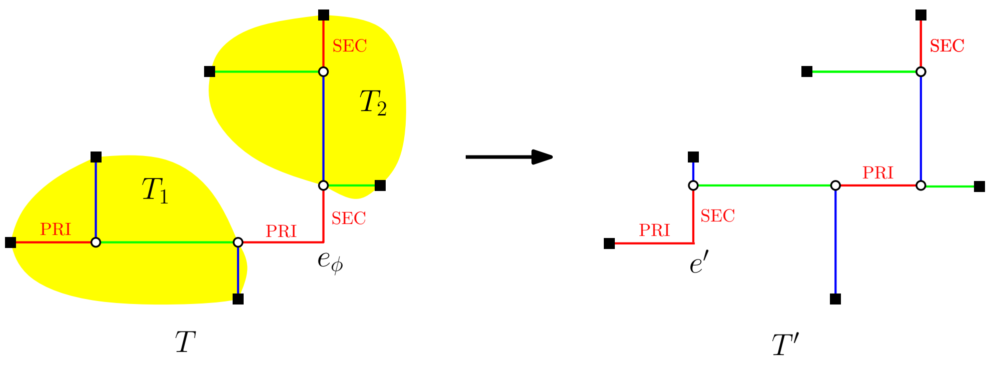


Figure 3.8: Illustration of proof of claim.

Second, assume that $e_\phi$ is a straight edge. Then all edges in $T$ are straight edges, and it follows from Lemma 3.7 that $T$ consists of *either* primary edges or secondary edges. We have obtained the following theorem.

**Theorem 3.8** *[211]* ***[Hwang form for rectilinear trees]*** *There exists a minimum rectilinear Steiner tree for a given set of terminals $N$ such that every full component has the so-called* Hwang form. *In the Hwang form, every full component spanning $k$ terminals consists of a complete corner (which is also referred to as the* backbone*) with terminal endpoints referred to as the* root *$t_1$ and the* tip *$t_k$. The leg containing the root is called the* long *leg and the leg containing the tip is called the* short *leg of the complete corner. There are two main types (i) and (ii) and two degenerate cases of type (i):*

- *Type (i) has $k-2$ alternating segments incident to the long leg and no segment incident to the short leg. Degenerate case (i′) has a zero-length short leg; i.e., the complete corner is degenerated into a complete line. Degenerate case (i″) is a cross interconnecting exactly four terminals.*

- *Type (ii) has $k-3$ alternating segments incident to the long leg and one segment incident to the short leg.*

Note that the terminology *short* leg and *long* leg is not meant to connote geometric length — rather, the long leg can have more incident segments than the short leg. The two main types are illustrated in Figure 3.9, and the two degenerate type (i) cases are depicted in Figure 3.10.

A non-degenerate Hwang form tree $T$ can be transformed into a *corner-flipped* version of itself (Figure 3.11). The corner-flipped version is obtained through a zero-shift (a series of slides and flips), or equivalently, by considering a canonical form where the edges are ordered in the *opposite* order to that of $T$.

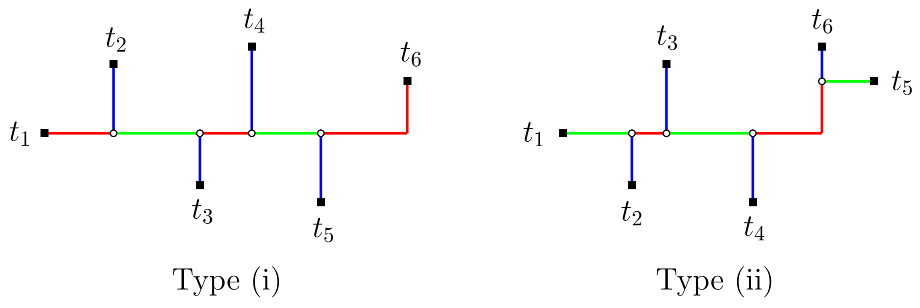


Figure 3.9: Hwang form full components: main types.

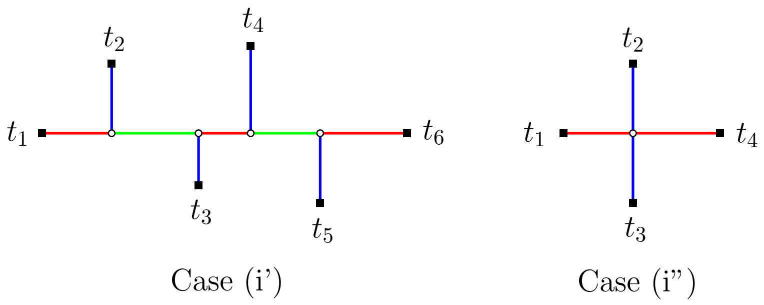


Figure 3.10: Hwang form full components: degenerate cases of type (i).

In the corner-flipped version the direction of the long leg from the corner point becomes the opposite to what it is in the original tree (east versus west or north versus south). This observation implies that we only need to consider *two* rather than four directions of the long leg when enumerating Hwang form trees.

The notion of corner-flipped trees creates three equivalence classes of Hwang forms given by the type and parity (odd/even) of the number of segments incident to the long leg. The first equivalence class contains Hwang form type (i) trees with an odd number of segments incident to the long leg, and type (ii) trees with an even number of incident segments. The second equivalence class contains Hwang form type (i) trees with an even number of segments incident to the long leg; and the third equivalence class contains type (ii) trees with an odd number of incident segments.

The transformation between corner-flipped Hwang forms implies the following lemma; the proof is left as Exercise 3.4:

**Lemma 3.9** ***[Short leg upper bound]*** *Let $T$ be a full and fulsome minimum rectilinear Steiner tree with Hwang form. For a type (i) tree, let $d$ denote the length of the short leg; otherwise, for a type (ii) tree, let $d$ denote the distance from the corner point to the Steiner point on the short leg. Let $st$ be any segment incident to the long leg and on the same side of the long leg as the short leg. Then $d < |st|$.*

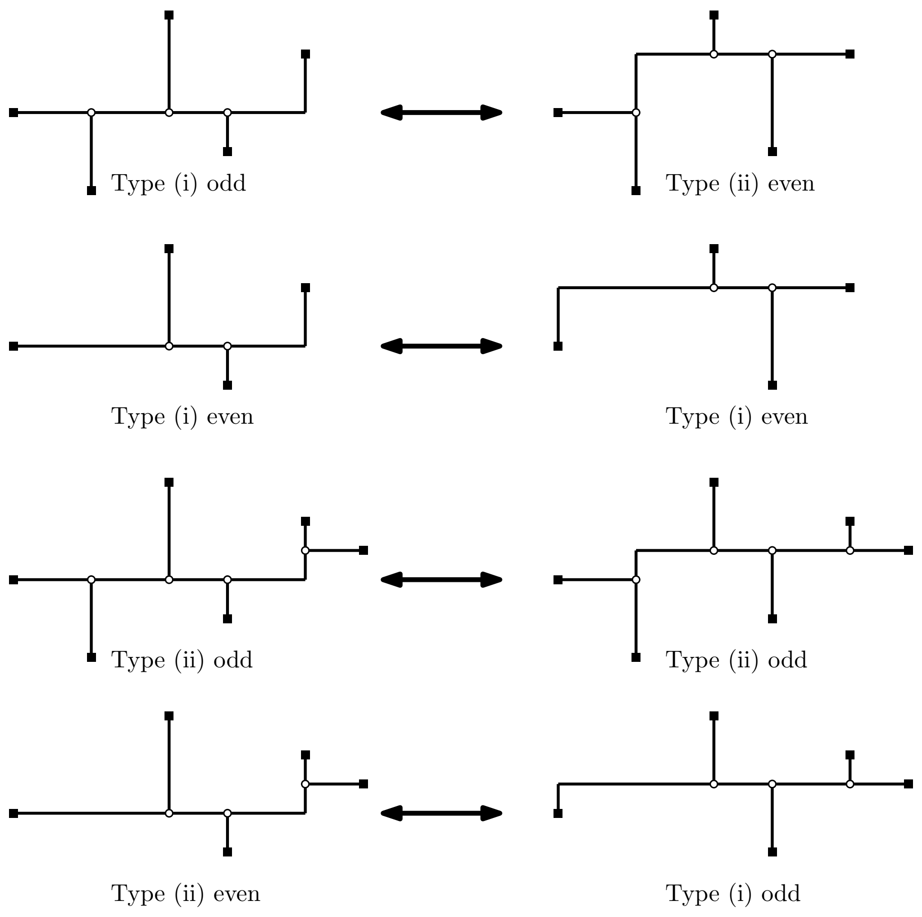


Figure 3.11: Corner-flipped Hwang forms.

Finally, we note that a Hwang form tree for a given topology can be constructed in linear time. That is, given a full Steiner topology with a caterpillar topology spanning a given set of terminals, we can locate the Steiner points according to a Hwang form — or decide that no such tree exists — in linear time in the number of terminals (see Exercise 3.5).

## 3.2 Global properties of minimum Steiner trees

In the previous section we looked at properties of Steiner points and full components, that is, local properties of minimum rectilinear Steiner trees. In this section we discuss some important global properties of these trees. First we define the related minimum spanning tree problem, and discuss the Steiner ratio — which is the smallest ratio between the length of a minimum Steiner tree and the length of a minimum spanning tree for the same set of terminals. We then define the Hanan grid, which gives a polynomial reduction of the rectilinear Steiner tree problem to the Steiner tree problem in graphs. We present a number of so-called empty regions associated with minimum rectilinear Steiner trees, and use these to give some bounds on the number of possible Hwang form trees. Finally, we show that the rectilinear Steiner tree problem is NP-hard, and that it is equivalent to any fixed orientation Steiner tree problem with two legal orientations.

### 3.2.1 Steiner ratio

Consider the problem of optimally interconnecting $N$ under the rectilinear metric *without* being allowed to use Steiner points. As in Chapter 1, this corresponds to computing a *minimum rectilinear spanning tree* for $N$. This problem can be solved in $O(n \log n)$ time, where $n = |N|$ [199].

Let $T(N)$ and $\overline{T}(N)$ denote a minimum rectilinear Steiner tree and a minimum rectilinear spanning tree, respectively, for $N$ under the rectilinear metric. Recall that we denote the rectilinear (or $\ell_1$) norm by $|\cdot|_1$. As in Section 2.5.1 in Chapter 2, define

$$\rho_2(N) := \frac{|T(N)|_1}{|\overline{T}(N)|_1}$$

to be the ratio between the lengths of a minimum rectilinear Steiner tree and a minimum rectilinear spanning tree for $N$. (Note that the notation '$\rho_2$' comes from the more general notation introduced in Section 2.5.1, where '$\rho_\lambda$' is the Steiner ratio in $\lambda$-geometry.) The *Steiner ratio* $\rho_2$ for the rectilinear metric is defined as

$$\rho_2 := \inf_N \rho_2(N).$$

In other words, the Steiner ratio is the smallest possible ratio between the minimum rectilinear Steiner tree and minimum rectilinear spanning tree lengths for any set of terminals.

Consider the set of terminals $N_4 = \{(-1,0),(0,-1),(1,0),(0,1)\}$. The minimum rectilinear Steiner tree for $N_4$ is a cross of length 4. Since the rectilinear distance between any pair of points in $N_4$ is 2, it follows that the length of a minimum rectilinear spanning tree for $N_4$ is 6; hence $\rho_2(N_4) = 4/6 = 2/3$. Thus, the Steiner ratio in the rectilinear metric is at most $2/3$. In the following theorem we show that the ratio is exactly $2/3$.

**Theorem 3.10** *[211] The Steiner ratio for the rectilinear plane is*

$$\rho_2 = \frac{2}{3}.$$

**Proof.** Let $T := T(N)$ and $\overline{T} := \overline{T}(N)$ denote a minimum rectilinear Steiner tree and a minimum rectilinear spanning tree, respectively, for some set of terminals $N$. We show that $|T|_1/|\overline{T}|_1 \geq 2/3$, or equivalently $|\overline{T}|_1 \leq 3/2|T|_1$.

The statement only needs to be established for every possible full component, in particular only for full components having the Hwang form. To see why, assume that $T$ is a union of full components $T_1, \ldots, T_m$. Assume that the Steiner ratio theorem holds for every full component $T_l$; then there exists a minimum rectilinear spanning tree, denoted by $\overline{T}_l$, for the set of terminals spanned by $T_l$ such that $|\overline{T}_l|_1 \leq 3/2|T_l|_1$. The union of $\overline{T}_1, \ldots, \overline{T}_m$, denoted by $\overline{T}'$, is clearly a spanning tree for $N$. Since

$$|\overline{T}|_1 \leq |\overline{T}'|_1 = \sum_{l=1}^{m} |\overline{T}_l|_1 \leq \sum_{l=1}^{m} 3/2|T_l|_1 = 3/2|T|_1,$$

the theorem also holds for any — not necessarily full — minimum rectilinear Steiner tree.

We therefore focus our attention on an arbitrary Hwang form full component $T_l$ spanning a set of terminals $N_l$, and show that $|\overline{T}_l|_1 \leq 3/2|T_l|_1$, where $\overline{T}_l$ is a minimum rectilinear spanning tree for $N_l$. Suppose $T_l$ spans $k = |N_l|$ terminals. Our proof will be by induction on $k$. The base case, $k \leq 4$, is left as Exercise 3.6.

First we assume that $T_l$ is a Hwang form type (i) tree. The root is denoted by $t_1$ and the alternating incident segments, in the direction from the root to the corner point, are denoted by $s_2t_2, \ldots, s_kt_k$, where $s_k$ is the corner point of $T_l$. It turns out to be useful also to consider the root $t_1$ as being connected to the long leg via a degenerate edge connecting to the Steiner point $s_1 = t_1$.

By converting $T_l$ to its corner-flipped form, if necessary, we can show that there exists an $i \in \{1, \ldots, k-3\}$ such that $|s_it_i| \leq |s_{i+2}t_{i+2}|$ and $|s_{i+1}t_{i+1}| \geq |s_{i+3}t_{i+3}|$ (see Exercise 3.7). Let $A = \{t_1, \ldots, t_i\}$ and $B = \{t_{i+3}, \ldots, t_k\}$ (Figure 3.12). Let $T_A$ and $T_B$ be the subtrees of $T_l$ that interconnect $A$ and $B$, respectively, and let $T_C$ be the subtree

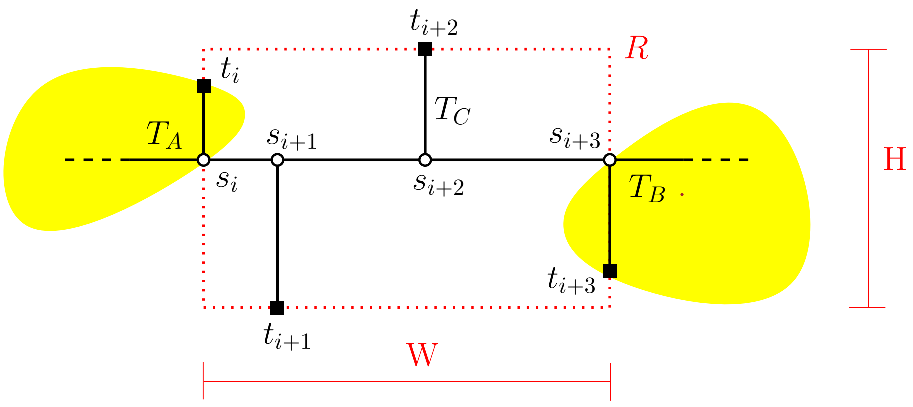


Figure 3.12: Illustration of proof of Theorem 3.10.

containing the remaining edges of $T_l$. Let $\overline{T}_A$ and $\overline{T}_B$ be minimum rectilinear spanning trees for $A$ and $B$, respectively. By the inductive hypothesis we have $|\overline{T}_A|_1 \leq 3/2|T_A|_1$ and $\overline{T}_B|_1 \leq 3/2|T_B|_1$.

Let $C = \{t_i, t_{i+1}, t_{i+2}, t_{i+3}\}$. Consider the boundary of the smallest axis-aligned rectangle $R$ that contains $C$. Rectangle $R$ has width $W = |s_i s_{i+3}|$ and height $H = |s_{i+1}t_{i+1}| + |s_{i+2}t_{i+2}|$, and we have $|T_C|_1 = W + H$. The boundary of $R$ has length $2(W + H)$, and it contains all terminals in $C$. Therefore, we can construct a spanning tree interconnecting $C$ (consisting of terminal-terminal connections only) by deleting the longest connection between two terminals on the boundary (this longest connection has length at least $(W + H)/2$). The length of a minimum rectilinear spanning tree $\overline{T}_C$ for $C$ is therefore bounded by

$$|\overline{T}_C|_1 \leq 3/2(W + H) = 3/2|T_C|_1.$$

In conclusion,

$$|\overline{T}_l|_1 \leq |\overline{T}_A|_1 + |\overline{T}_B|_1 + |\overline{T}_C|_1 \leq 3/2(|T_A|_1 + |T_B|_1 + |T_C|_1) = 3/2|T_l|_1.$$

For a Hwang form type (ii) tree all the arguments above can be repeated; the single terminal attached to the short leg will never be part of the set $C$. ∎

### 3.2.2 Hanan grid reduction

One of the first and most important structural results for the rectilinear Steiner tree problem was given by Hanan [189] in 1966. We begin with a definition of the Hanan grid and the Hanan grid graph.

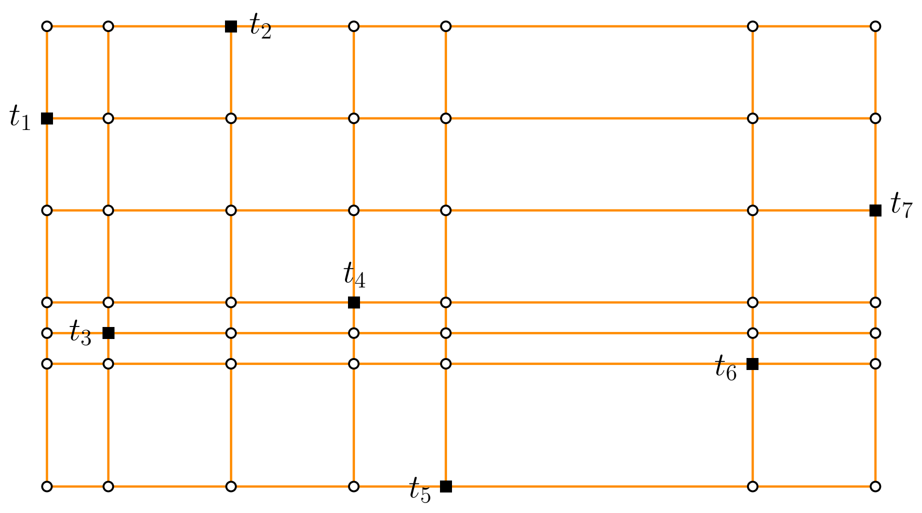


Figure 3.13: Hanan grid graph for the set of terminals given in Figure 3.1.

**Definitions [Hanan grid, Hanan grid graph]**: The *Hanan grid* $\mathbf{GG}(N)$ is the set of intersection points obtained by drawing a horizontal line and a vertical line through each point in $N$; note that $N \subseteq \mathbf{GG}(N)$. The *Hanan grid graph* is a geometric network $\mathbf{H}(N)$ that has $\mathbf{GG}(N)$ as its vertex set; there is an edge between two vertices $u, v \in \mathbf{GG}(N)$ if $u$ and $v$ are adjacent along a horizontal or vertical line, and the weight of edge $(u, v)$ is the Euclidean distance $|uv|$ between $u$ and $v$).

The Hanan grid graph for the set of terminals given in Figure 3.1 is shown in Figure 3.13. The following theorem is an immediate corollary of Theorem 3.8:

**Theorem 3.11** *[189]* ***[Hanan grid reduction]*** *There exists a minimum rectilinear Steiner tree for $N$ such that every Steiner point belongs to $\mathbf{GG}(N)$.*

One obvious consequence of this theorem is that we only need to consider a polynomial number of Steiner point candidates — namely the $O(n^2)$ points in the Hanan grid. Also, in terms of computational complexity, this means that there exist short certificates of optimal solutions, as we only need to consider Steiner point coordinates that are amongst the coordinates of the given terminals. The rectilinear Steiner tree problem is therefore in NP — in contrast with the Euclidean Steiner tree problem, for which this question is still unsettled (see Section 1.3.3 in Chapter 1).

The Steiner tree problem in an edge-weighted graph $G = (V, E)$ is the problem of constructing a tree in $G$ interconnecting a given set of terminals $N_G \subseteq V$ with minimum weight. It follows from the Hanan grid reduction that the rectilinear Steiner tree problem is equivalent to solving the Steiner tree problem in the Hanan grid graph $\mathbf{H}(N)$ with terminal set $N$. The rectilinear Steiner tree problem can therefore be solved as a planar graph problem with at most $n^2$ vertices and $2n(n-1)$ edges, where $n = |N|$. An overview of structural properties and exact algorithms for the Steiner tree problem in graphs is given in Section 5.1 in Chapter 5. In the remainder of this section we cover some specialised algorithms that can be applied to the Hanan grid graph.

### Graph reductions for the Hanan grid graph

For particular problem instances, so-called *graph reductions* can often significantly reduce the size of the Hanan grid graph $\mathbf{H}(N)$ while retaining at least one minimum Steiner tree for $\mathbf{H}(N)$ in the reduced graph. Graph reductions iteratively remove vertices and edges from the graph using a series of *reduction tests*. Here we briefly mention some of the specialised reductions that are effective for the Hanan grid graph; more general graph reductions are discussed in Section 5.1 in Chapter 5.

Provan [320, 321] introduced the *path-convex hull*, which is a generalisation of the convex hull for planar point sets to vertex sets in planar graphs. Consider a straight-line embedding of an edge-weighted planar graph $G = (V, E)$. Let $w$ be a closed walk in $G$ — possibly traversing some edges in $G$ more than once — and let $R(w)$ be the polygonal region defined by $w$. Let $c(w)$ be the total weight of the edges travelled by $w$, that is, the *perimeter* of the polygonal region $R(w)$.

**Definition [Path-convex hull]**: A polygonal region $R(w)$ given by a walk $w$ in a straight-line embedding of a planar graph $G = (V, E)$ with terminal set $N_G \subseteq V$ is a *path-convex hull* if all terminals $N_G$ are within $R(w)$ and the perimeter $c(w)$ is minimum amongst all such walks.

**Theorem 3.12** *[320] Let $R(w)$ be a path-convex hull for a straight-line embedding of a planar graph $G = (V, E)$ with terminal set $N_G \subseteq V$. Then there exists a minimum Steiner tree for $N_G$ that lies entirely in $R(w)$.*

Let us apply this theorem to the Hanan grid graph in Figure 3.13. The walk $w_1$ along the outer boundary of the Hanan grid graph — or the boundary of the smallest axis-aligned rectangle that contains $N$ — clearly defines a region $R(w_1)$ that is a path-convex hull. The walk $w_1$ visits the terminals $t_1$, $t_2$, $t_7$ and $t_5$ in clockwise order around the Hanan grid graph. If the path from $t_1$ to $t_2$ instead of moving up and right, first moved right and then up, essentially ’cutting off’ the leftmost upper rectangle in the Hanan grid graph, the new walk $w_2$ would still define a path-convex hull. By iteratively removing rectangles in the Hanan grid, we obtain a series of smaller path-convex hulls, each guaranteed to contain at least one minimum rectilinear Steiner tree. For the Hanan grid graph in Figure 3.13,

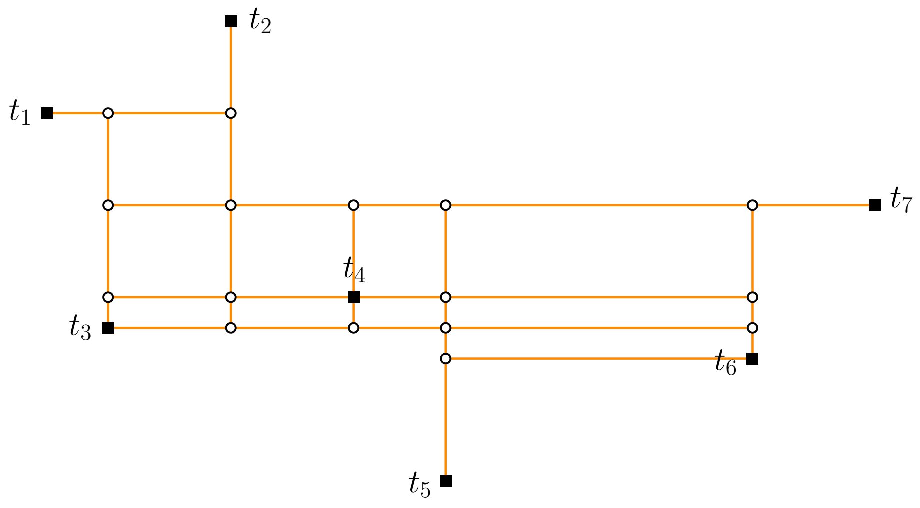


Figure 3.14: Reduced Hanan grid graph based on rectangle removal.

the final result would be the reduced Hanan grid graph shown in Figure 3.14, which is significantly smaller than the original Hanan grid graph.

Winter [406] proposed several reduction tests that take advantage of the special structure of the Hanan grid graph, in particular that vertices have low degree and that many edges have the same length. Uchoa, Poggi de Aragão and Ribeiro [375] extended the ideas of Winter to reducing Hanan grid graphs with holes/obstacles (see also Section 4.2 in Chapter 4).

Another straightforward and fast method to reduce the Hanan grid graph is to enumerate Hwang form full components (see Section 3.3). That is, take the set of generated full Steiner trees (FSTs) and place them on the Hanan grid. (Note that every Hwang form full component is contained in the Hanan grid.) Edges and Steiner points in the Hanan grid which are not used by any FST can be deleted. This is in practice the fastest way to reduce the Hanan grid. The number of Steiner points that remain is almost linear in the number of terminals; for problem instances with 1000 terminals, less than 0.5% of the vertices in the Hanan grid remain after FST generation [429], and the FST generation only takes a fraction of a second on a modern computer.

We note that a number of generalisations of the rectilinear Steiner tree problem can be solved in the underlying Hanan grid. Ganley and Cohoon [164] have shown that the rectilinear Steiner tree problem with rectilinear obstacles can be solved in the Hanan grid given by the terminals and the corners of the obstacles; see Section 4.2.3 of Chapter 4. Zachariasen [430] has presented a catalog of problems that have an optimal solution in the Hanan grid, including so-called weighted-obstacle, group and prize-collecting variants.

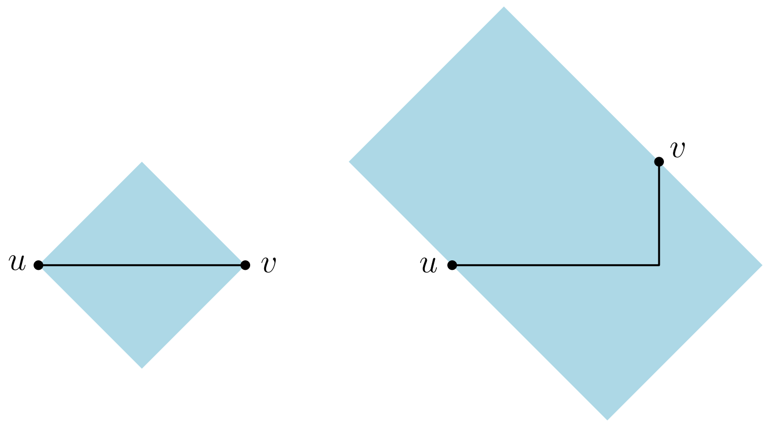


Figure 3.15: Empty lunes for a straight edge (left) and a bent edge (right).

### 3.2.3 Empty regions

In this section we study some necessary geometric conditions that must be satisfied by minimum rectilinear Steiner trees for a given terminal set $N$, independently of the topology of the tree. All the conditions presented are so-called *empty region* properties. An empty region is a region in the plane that can be shown to be free of Steiner points and/or terminals if certain conditions are fulfilled. All the empty regions can be efficiently computed without having to first compute a minimum rectilinear Steiner tree, and they are therefore useful as efficient pruning conditions for eliminating non-feasible full components; furthermore, they are useful from a theoretical viewpoint for helping bound the number of candidate full components, as will be shown in the next section.

#### The lune property

Recall from Chapter 1 that a lune $\mathcal{L}(u,v)$ is defined as the set of points that are strictly within distance $|uv|_1$ of both $u$ and $v$ (where distance here is given by the rectilinear metric). If $(u,v)$ is an edge in a minimum rectilinear Steiner tree, then $\mathcal{L}(u,v)$ does not contain any points of the tree that do not lie on $(u,v)$ (Lemma 1.13). Geometrically, a lune for edge $(u,v)$ is the intersection of the interiors of the two $\ell_1$ circles with radius $|uv|_1$ centred at $u$ and $v$, respectively (Figure 3.15).

Consider a full component $T$ in a minimum rectilinear Steiner tree. Not only are the lunes empty; they are also pairwise geometrically disjoint:

**Lemma 3.13** *[413]* ***[Disjoint lunes property]*** *Let $T$ be a full and fulsome minimum rectilinear Steiner tree with Hwang form. For any pair of distinct edges $(u,v)$ and $(w,z)$ in $T$, we have $\mathcal{L}(u,v) \cap \mathcal{L}(w,z) = \emptyset$.*

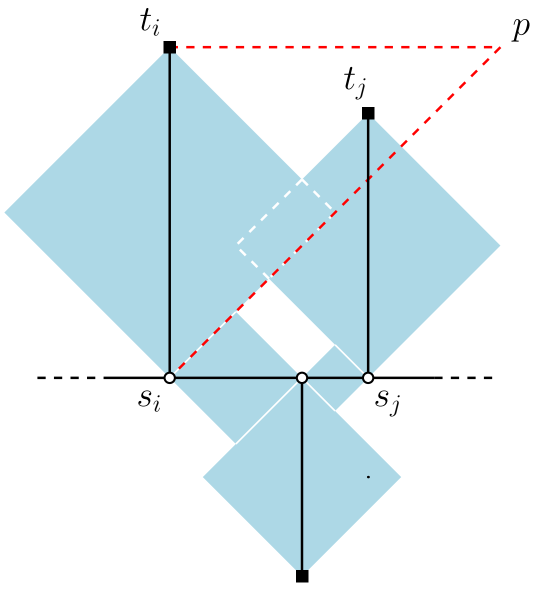


Figure 3.16: Illustration of proof of Lemma 3.13.

**Proof.** Consider any pair of distinct edges from $T$. If one of the edges is part of the backbone of $T$, then the corresponding lunes are clearly disjoint. If the two edges are not on the same side of the backbone, then they are also disjoint.

The only remaining case is when the edges are on the same side of the backbone. Let $s_it_i$ and $s_jt_j$ be a pair of such incident segments; we assume without loss of generality that $s_it_i$ and $s_jt_j$ are both vertical segments and that $|s_it_i| \geq |s_jt_j|$.

Suppose that $\mathcal{L}(s_i, t_i) \cap \mathcal{L}(s_j, t_j) \neq \emptyset$. Then we have the situation depicted in Figure 3.16: let $p$ be the point on the horizontal line through $t_i$ on the same side of $s_it_i$ as $t_j$ such that $|t_ip| = |t_is_i|$. Since the lunes overlap, it follows that $t_j$ is in the interior of triangle $\triangle s_it_ip$, which implies that $|s_is_j| < |s_jt_j|$. We can now construct a shorter tree by removing segment $s_jt_j$ and adding a connection of length $|s_is_j|$ from $t_j$ to segment $s_it_i$, contradicting the optimality of $T$. ▮

The proof of Lemma 3.13 implies that incident segments on the same side of the backbone of a Hwang form full component cannot be too close to each other. More precisely, if $s_it_i$ and $s_jt_j$ are two incident segments, then $|s_is_j| \geq \min(|s_it_i|, |s_jt_j|)$.

### The empty rectangle property

Consider two perpendicular segments $uw$ and $wv$ meeting at a node $w$ (Figure 3.17). Let $\mathcal{R}(u, v)$ be the interior of the axis-aligned rectangle with sides $uw$ and $wv$.

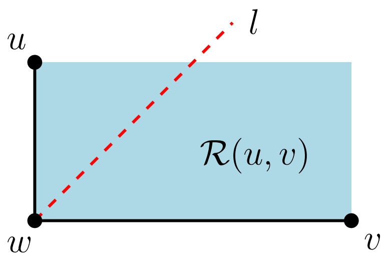


Figure 3.17: Illustration of proof of Lemma 3.14.

**Lemma 3.14** ***[Empty rectangle property]*** *If $uw$ and $wv$ are perpendicular segments in a minimum rectilinear Steiner tree $T$, then $\mathcal{R}(u,v)$ contains no point of $T$.*

**Proof.** Assume on the contrary that $T$ contains a point $p \in \mathcal{R}(u,v)$. Let $l$ be the line through $w$ which bisects the perpendicular angle, and assume that $p$ is above $l$ (in Figure 3.17). Remove $uw$ from $T$, splitting $T$ into two connected components. If $p$ belongs to the same component as $u$, then add a vertical segment from $p$ down to segment $wv$, otherwise reconnect by connecting $u$ and $p$. In both cases the tree is shortened, a contradiction. If $p$ is below $l$ a similar argument shows that the tree also can be shortened in this case. Finally, assume that $p$ is exactly on the line $l$. Since $T$ consists of vertical and horizontal segments, $T$ must contain another point $p' \in \mathcal{R}(u,v)$ that is either above or below $l$, again allowing us to shorten the tree. ∎

The empty rectangle property given in Lemma 3.14 has been used with great success in the design of both exact and heuristic methods for the rectilinear Steiner tree problem [27, 274, 429].

We conclude this section with another empty region denoted the *empty inner rectangle property*. (The proof of the lemma is left as Exercise 3.8.)

**Lemma 3.15** *Let $T'$ be a fulsome minimum rectilinear Steiner tree, and let $T$ be a full component of $T'$ with Hwang form. Let $t_1$ be the root of $T$. If $T$ is a type (i) full component, then let $t_k$ be the tip of $T$; otherwise, let $t_k$ be the single terminal attached to the short leg of $T$. Then $\mathcal{R}(t_1, t_k)$ contains no point of $T'$ (other than the edges of $T$).*

### 3.2.4 Bounds on the number of full components

In the previous sections we described some necessary structural properties of minimum rectilinear Steiner trees. Each of the full components of a minimum rectilinear Steiner tree $T$ must fulfil these necessary properties. This leads to the following question: Assuming that $T$ is *unknown*, how many full components having the Hwang form — spanning a subset of terminals and fulfilling some subset of necessary properties — can be constructed?

Bounds on the number of such full components are highly relevant, e.g., for the GeoSteiner algorithm that enumerates full components in the so-called generation phase of the algorithm (see Section 3.3).

A trivial upper bound on the number of candidate full components is $O(2^n)$. To see why, consider some subset of terminals $S \subseteq N$; there exist at most four trees for $S$ having the Hwang form — one for each of the four root candidates having a minimum or maximum $x$- or $y$-coordinate in $S$.

It turns out that all worst-case bounds are in fact *exponential*, and remain so no matter what (known) necessary structural properties are enforced. The good news is that the *expected* number of full components is much smaller. Assume that the terminals are randomly and uniformly distributed in a square, i.e., each terminal coordinate is chosen uniformly at random within an interval. Then the expected number of full components spanning $k$ terminals (and fulfilling the empty and disjoint lune properties) is $O(n\pi^{k-1})$, that is, *linear* in $n$. In the following subsections we first present worst-case upper and lower bounds on the number of full components, and then we prove the probabilistic bound on the expected number of full components.

### Worst-case upper and lower bounds

In this section we consider worst-case upper and lower bounds on the number of full components. We focus on one relatively simple upper bound of $O(n \cdot 1.42^n)$, given by Fößmeier and Kaufmann [156], that improves on the trivial $O(2^n)$ bound. The presented bound is not the best known upper bound, but the proof of the bound gives the flavour of the arguments that are used to prove such bounds. The best known worst-case bound is by Fuchs et al. [160], who have shown that the number of full components satisfying the so-called *tree star* property is $O^*(1.357^n)$, where the $O^*$ notation indicates that factors of polynomial order in $n$ are suppressed.

The upper bound is achieved by requiring that all full components satisfy the empty rectangle property (Lemma 3.14). Furthermore, in order to simplify the arguments, we assume that the terminals are in general position (that is, have pairwise different $x$- and $y$-coordinates).

Assume that such a set of terminals $N$ is given, and that we wish to construct a potential full component $T$ of a minimum rectilinear Steiner tree for $N$ spanning $k$ terminals. By Section 3.1.2, we can assume that $T$ has Hwang form. We also assume that the root of $T$ and the orientation of the long leg of $T$ are given. The $k$ terminals of $T$ are labelled $t_1, t_2, \ldots, t_k$ in order along the backbone of the Hwang form tree, where terminal $t_1$ is the root of the tree (as illustrated in Figure 3.18).

With $T$ defined in this way, we have the following lemma.

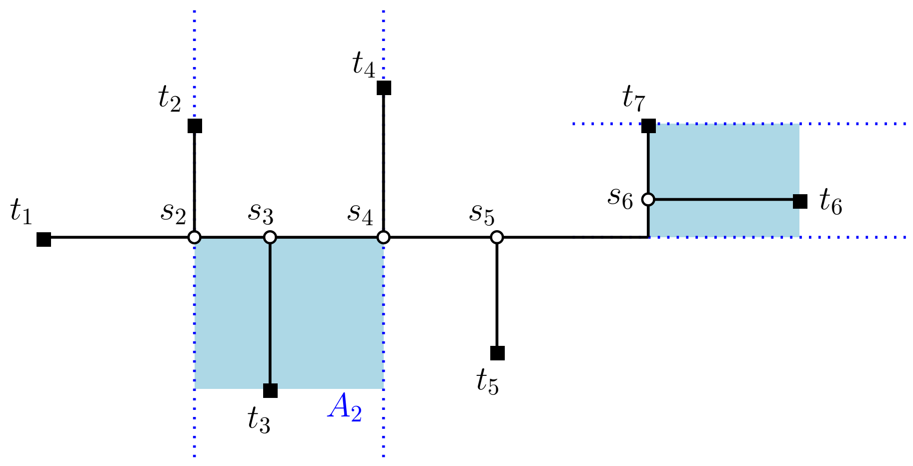


Figure 3.18: Type (ii) full component with the Hwang form satisfying the empty rectangle property. Terminal $t_3$ makes empty rectangles with Steiner point $s_2$ and Steiner point $s_4$. Lemma 3.16 shows that if the root of the component and terminals $t_2$ and $t_4$ are given, terminal $t_3$ is unique. The same holds for terminal $t_6$ if the root and the tip of the full component are given.

**Lemma 3.16** *If two consecutive terminals $t_i$ and $t_{i+2}$ on the same side of the long leg of $T$ are given, then the terminal $t_{i+1}$ on the opposite side of the long leg is* uniquely *determined by $t_i$ and $t_{i+2}$. The same holds for the terminal $t_{k-1}$ attached to the short leg if the tip of the full component is given (for a type (ii) full component).*

**Proof.** We can assume, without loss of generality, that the root $t_1$ is the leftmost terminal in the full component (and hence the long leg is horizontal), and that terminals $t_i$ and $t_{i+2}$ are above the long leg (as in Figure 3.18). Consider the infinite vertical strip given by terminals $t_i$ and $t_{i+2}$, and let $A_i$ be the intersection of the strip with the half-plane below the long leg. Clearly, terminal $t_{i+1}$ must be located in $A_i$.

Now, since we assume that $T$ satisfies the empty rectangle property, terminal $t_{i+1}$ must necessarily be the terminal in $A_i$ with the largest $y$-coordinate — otherwise one of the two rectangles given by $t_{i+1}$ and its adjacent Steiner point would not be free of terminals. Also, since the terminals are assumed to be in general position, $t_{i+1}$ is unique.

The same arguments can be used to prove that the terminal $t_{k-1}$ attached to the short leg is unique. Here $t_{k-1}$ is the unique terminal with the smallest $x$-coordinate to the right of the tip, and within the horizontal strip given by the root and the tip. ∎

We now ask the question: How many full components spanning $k$ terminals can be constructed for a given terminal $t$, assuming that $t$ is the root and the leftmost terminal of the full component? Let $n(t)$ be the total number of terminals in $N$ to the right of $t$, and assume, without loss of generality, that at most $\lceil n(t)/2 \rceil$ terminals are above and to the right of $t$ (otherwise we change 'above' to 'below' in the following discussion). We consider four cases.

**Case 1: $T$ is a type (i) full component and the tip is above the long leg**
Assume that the terminals in $T$ above the long leg are given. Lemma 3.16 implies that there exists *at most one* full component for this given set of terminals. Since at most $\lfloor k/2 \rfloor$ terminals are above the long leg, the total number of possible full components is at most $\binom{\lceil n(t)/2 \rceil}{\lfloor k/2 \rfloor}$.

**Case 2: $T$ is a type (i) full component and the tip is below the long leg**
Assume that the terminals in $T$ above the long leg and the tip of $T$ are given. Again, there exists at most one full component for this given set of terminals. The total number of possible full components is at most $n(t) \cdot \binom{\lceil n(t)/2 \rceil}{\lfloor k/2 \rfloor}$ (since there are up to $n(t)$ choices for the tip).

**Case 3: $T$ is a type (ii) full component and the tip is above the long leg**
Assume that the terminals in $T$ above the long leg, except the terminal attached to the short leg, are given. By Lemma 3.16, there exists at most one full component for this given set of terminals. The total number of possible full components is at most $\binom{\lceil n(t)/2 \rceil}{\lfloor k/2 \rfloor}$.

**Case 4: $T$ is a type (ii) full component and the tip is below the long leg**
Assume that the terminals in $T$ above the long leg and the tip of $T$ are given. Again, by Lemma 3.16, there exists at most one full component for this given set of terminals. The total number of possible full components is at most $n(t) \cdot \binom{\lceil n(t)/2 \rceil}{\lfloor k/2 \rfloor}$.

In summary, for a given $k$, where $1 \leq k-1 \leq n(t)$, the number of full components is bounded by $n(t) \cdot \binom{\lceil n(t)/2 \rceil}{\lfloor k/2 \rfloor}$. Taken over all possible values of $k$, the number of full components is bounded by:

$$\sum_{k=2}^{n(t)+1} n(t) \cdot \binom{\lceil n(t)/2 \rceil}{\lfloor k/2 \rfloor} \leq 2 \cdot n \cdot \sum_{k=1}^{\lceil n(t)/2 \rceil} \binom{\lceil n(t)/2 \rceil}{k} \leq 2 \cdot n \cdot 2^{\lceil n(t)/2 \rceil + 1} \leq 4 \cdot n \cdot 2^{\lceil n(t)/2 \rceil}.$$

Let us order the terminals of the problem instance from left to right (that is, by their $x$-coordinate). The leftmost terminal has $n-1$ terminals to the right, the next has $n-2$ terminals to the right and so on. Taken over all root candidates, this gives the following bound on the number of full components:

$$4 \cdot n \cdot (2^{\lceil (n-1)/2 \rceil} + 2^{\lceil (n-2)/2 \rceil} + \ldots + 2^{\lceil 0/2 \rceil}) \leq 16 \cdot n \cdot 2^{\lceil n/2 \rceil}.$$

Finally, there are four orientations of the long leg, resulting in the bound $64 \cdot n \cdot 2^{\lceil n/2 \rceil}$, which is $O(n \cdot 1.42^n)$, implying the following theorem.

**Theorem 3.17** *[156] Given a set $N$ of $n$ terminals in general position, the number of full* *components with Hwang form, and satisfying the empty rectangle property, is* $O(n{\cdot}1.42^n)$.

Fößmeier and Kaufmann [156] also give an exponential worst-case lower bound on the number of full components. More precisely, they show that the problem instance illustrated in Figure 3.19 has an exponential number of full components that satisfy the so-called *tree star* property — a strictly stronger property than the empty rectangle property. In the constructed problem instance, if terminals $t_2$ and $t_{12}$ are part of a full component, then any combination of the remaining terminals above the long leg $t_4$, $t_6$, $t_8$ and $t_{10}$ results in a full component that satisfies the tree star property. Since the problem instance can be infinitely extended (see Figure 3.19), the number of full components is $2^{4/5 \cdot n/2}$, which is $\Omega(1.32^n)$. Even if more necessary conditions are enforced, experiments by Zachariasen [429] show that the number of full components for this particular series of problem instances grows as $\Omega(1.06^n)$.

**Probabilistic bounds**

In this section we present the best bound known for the expected number of full components.[36] This bound was given by Wulff-Nilsen [413] in the theorem below, and it makes use of the empty lune and disjoint lune properties form Section 3.2.3.

**Theorem 3.18** *[413] Given a set $N$ of $n$ terminals randomly and uniformly distributed in a square, the expected number of full components with the Hwang form spanning $k$ terminals, and satisfying the empty lune property and the disjoint lune property, is* $O(n\pi^{k-1})$.

**Proof.** Let $\mathcal{U}$ be a square in the plane, and assume that the $n$ terminals are randomly and uniformly distributed in $\mathcal{U}$. Let $T$ be some full component having the Hwang form and spanning $k$ terminals, $2 \leq k \leq n$. We assume that $T$ satisfies the empty lune property and disjoint lune property (Lemma 3.13). The $k$ terminals of $T$ are denoted $t_1, t_2, \ldots, t_k$ along the backbone of the Hwang form tree. Terminal $t_1$ is the root of the tree, and $t_k$ is the tip of the tree. For a type (ii) tree, $t_{k-1}$ is the single terminal attached to the short leg of the tree.

We consider a process where the terminals are selected in order $t_1, t_2, \ldots, t_k$, and give a bound on the expected number of choices in each of the $k$ steps. (For a type (ii) tree, the selection order of the last two terminals is $t_k$ and then $t_{k-1}$.)

In the first step we choose $t_1$, a direction of the long leg (north, south, east or west) and a side (left or right of the long leg as seen from $t_1$ in the given direction) where the second terminal $t_2$ should be chosen. Clearly, there are exactly $8n$ choices in this first step.

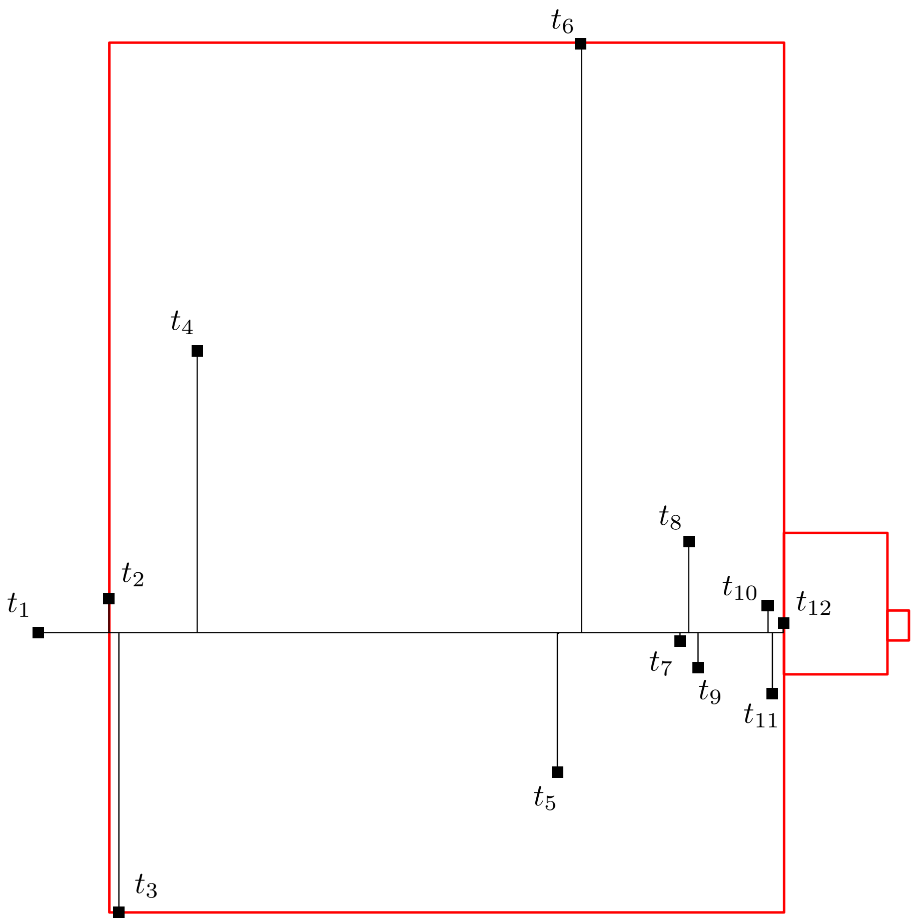


Figure 3.19: A series of problem instances with an exponential number of full components. The small rectangles are scaled-down versions of the large rectangle. The series of instances is constructed by iteratively adding scaled-down configurations of terminals $t_3$–$t_{12}$, where terminal $t_{12}$ acts as terminal $t_2$ in the next scaled-down configuration.

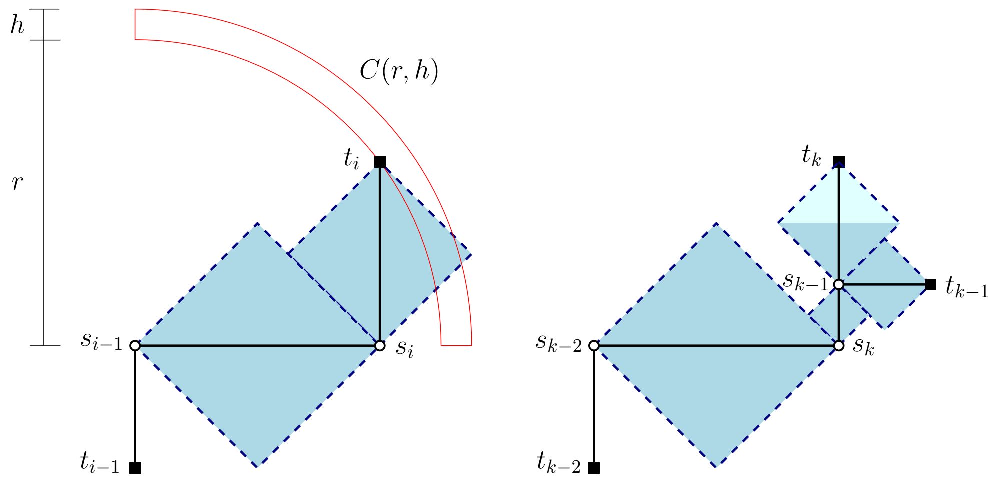


Figure 3.20: Illustration for Cases 1 and 2 (left) and Cases 3 and 4 (right).

Assume that we have chosen the first $i-1$ terminals, $i \geq 2$. Let $\mathcal{L}_i$ be the union of the lunes of the (partial) tree given by the terminals $t_1, \ldots, t_{i-1}$. Note that $\mathcal{L}_i$ must be completely free of terminals; therefore, the remaining $n-i+1$ terminals of $N$ are randomly and uniformly distributed in $\mathcal{U} \setminus \mathcal{L}_i$. We denote by $A$ the area of $\mathcal{U} \setminus \mathcal{L}_i$.

We distinguish between four cases, depending on the type of full component that is constructed and the relation between $i$ and $k$. An *inner* terminal of a Hwang form tree is a terminal that is attached to the long leg of the tree, not including the root or the tip of the tree.

**Case 1: Terminal $t_i$ is an inner terminal**
Consider the terminal $t_{i-1}$ and the Steiner point $s_{i-1}$ connecting $t_{i-1}$ to the long leg of $T$. Assume without loss of generality that $s_{i-1}$ is located at the origin, and that a candidate terminal $t_i = (x, y)$ is located somewhere in the first quadrant; the corresponding Steiner point $s_i = (x, 0)$ is located on the $x$-axis (Figure 3.20, left). We would like to bound the expected number of choices of $t_i$.

The lunes $\mathcal{L}(s_{i-1}, s_i)$ and $\mathcal{L}(s_i, t_i)$ should contain no terminals in their interior. The combined area of these two disjoint lunes is $\frac{1}{2}x^2 + \frac{1}{2}y^2 = \frac{1}{2}r^2$, where $r$ is the Euclidean distance from $t_i$ to the origin. Clearly, at least half of the two lunes are within the given square $\mathcal{U}$. Thus, in order for $t_i$ to be a candidate terminal, an area of size at least $\frac{1}{4}r^2$ (out of total area $A$) must be free of terminals. The probability that this is the case is no more than

$$\left(1 - \frac{r^2}{4A}\right)^{n-i}$$

since each of the remaining $n-i$ terminals must be chosen from the area uncovered by the lunes. Note that we must have $1 - r^2/(4A) > 0$, so $2\sqrt{A}$ is an upper bound on $r$.

Define $C(r,h)$ to be the region in the first quadrant containing the points that are at least distance $r$ and at most distance $r+h$ from the origin (Figure 3.20, left). Region $C(r,h)$ has area

$$\frac{\pi}{4}((r+h)^2 - r^2) = \frac{\pi}{2}rh + \frac{\pi}{4}h^2,$$

and the expected number of terminals in $C(r,h)$ is at most

$$(n-i+1)\left(\frac{\pi}{2}rh + \frac{\pi}{4}h^2\right)/A.$$

The expected number of candidate terminals for $t_i$ in $C(r,h)$ is bounded by

$$\left(1-\frac{r^2}{4A}\right)^{n-i}(n-i+1)\left(\frac{\pi}{2}rh + \frac{\pi}{4}h^2\right)/A.$$

By integrating over $r$ (and taking the limit as $h \to 0$), we obtain a bound $E$ on the expected number of candidates for $t_i$:

$$\begin{aligned} E &\leq \int_{r=0}^{2\sqrt{A}} \left(1-\frac{r^2}{4A}\right)^{n-i}(n-i+1)\frac{\pi}{2A}rdr \\ &= \frac{\pi(n-i+1)}{2A}\int_{r=0}^{2\sqrt{A}} \left(1-\frac{r^2}{4A}\right)^{n-i} rdr \\ &= \frac{\pi(n-i+1)}{2A}\left[-2A\frac{\left(1-\frac{r^2}{4A}\right)^{n-i+1}}{n-i+1}\right]_{r=0}^{2\sqrt{A}} \\ &= \pi \end{aligned}$$

**Case 2: Terminal $t_i$ is the tip of a type (i) full component ($i=k$)**
This case is identical to Case 1 except that the connection point $s_i$ from $t_i$ to the long leg is a corner point rather than a Steiner point. However, this does not change the bound, as the same types of lunes as in Case 1 must be free of terminals. The expected number of candidates is bounded by $\pi$.

**Case 3: Terminal $t_i$ is the tip of a type (ii) full component ($i=k$)**
This case is similar to Case 2, but with a crucial difference. Since an *unknown* terminal $t_{k-1}$ will be attached through a Steiner point $s_{k-1}$ to the short leg, only the lunes $\mathcal{L}(s_k, s_{k-1})$ and $\mathcal{L}(s_{k-1}, t_k)$ — rather than $\mathcal{L}(s_k, t_k)$ — can be assumed to be free of terminals (Figure 3.20, right). However, lunes $\mathcal{L}(s_k, s_{k-1})$ and $\mathcal{L}(s_{k-1}, t_k)$ have an area that is at least one half of that of $\mathcal{L}(s_k, t_k)$. Letting $r$ be the Euclidean distance from $s_{k-2}$ to $t_k$, it follows from the arguments of Case 1 that the expected number of candidates is bounded by $2\pi$.

In order to 'prepare' for the handling of Case 4, we will assume that only *half* of the lune $\mathcal{L}(s_{k-1}, t_k)$ (say, the half that is closest to $s_{k-1}$) is free of terminals. This increases the bound with the same factor, so it becomes $4\pi$.

**Case 4: Terminal $t_i$ is the terminal attached to the short leg for a type (ii) full component ($i = k - 1$)**
In this case we consider $r$ to be the Euclidean distance from $t_k$ to $t_{k-1}$. The lunes $\mathcal{L}(s_{k-1}, t_k)$ and $\mathcal{L}(s_{k-1}, t_{k-1})$ are free of terminals, so the arguments are in principle similar to those of Case 1. However, in order to obtain independence of expectation, we need to consider empty regions that are disjoint from the empty regions that are used for terminals $t_1, \dots, t_{k-2}$ and $t_k$. In Case 3, lune $\mathcal{L}(s_{k-1}, t_k)$ was also used — but only half of it. The other half was reserved for this case, resulting in a bound of $2\pi$ for the expected number of candidates.

In conclusion, the expected number of type (i) full components is bounded by $(8n)\pi^{k-1}$; similarly, the expected number of type (ii) full components is bounded by $(8n)\pi^{k-3}(4\pi)(2\pi)$. In both cases we obtain a bound of $O(n\pi^{k-1})$. ∎

The theorem shows that the expected number of full components spanning $k$ terminals is *linear* in $n$. However, the theorem does not provide a polynomial bound in $n$ on the total number of expected full components (over all $k$); experimental evidence indicates that the expected total number of full components fulfilling the empty lune and disjoint lune properties is *not* polynomial in $n$ [413]. Thus, in order to obtain a better bound, additional necessary properties such as the bottleneck distance property must be enforced.

### 3.2.5 Computational complexity

In this section we show that the rectilinear Steiner tree problem is NP-hard. More precisely, we prove that the following decision problem is NP-complete:

RECTILINEAR STEINER TREE DECISION PROBLEM
**Instance**: A finite set $N$ of points with integer coordinates in the plane, and a positive integer $L$.
**Question**: Is there a rectilinear Steiner tree $T$ with terminal set $N$ such that the length of $T$ is at most $L$?

In contrast to the Euclidean Steiner tree problem (see Chapter 1), we are not able to show that the restricted problem where the terminals are lying on two parallel lines is NP-complete; in fact, this problem is polynomially solvable for the rectilinear problem, as will be shown in Section 3.5. This is despite the fact that the corresponding $\lambda$-geometry problem is NP-complete for points lying on two parallel lines for $\lambda \geq 3$ (see Chapter 2). The rectilinear problem, which is identical to the $\lambda$-geometry problem for $\lambda = 2$, is therefore a limiting case when it comes to computational complexity.

On the positive side, the rectilinear Steiner tree decision problem is clearly in NP; since the Steiner points can be assumed to be on the Hanan grid for $N$, their $x$- and $y$-coordinates can be assumed to be identical to the integer coordinates of the terminals. If the set of Steiner points of a rectilinear Steiner tree for $N$ is known, then the length of this tree can be obtained in polynomial time by computing a minimum spanning tree of $N$ and the given Steiner points.

The proof of NP-completeness given by Garey and Johnson [170] in 1977 involves a series of polynomial-time reductions, and in this section we only describe the last reduction — which is also the only reduction that involves planar geometry. Recall that a *vertex cover* of a graph $G = (V, E)$ is a subset of vertices $C \subseteq V$ that contains at least one endpoint of every edge $e \in E$. Garey and Johnson first proved that the following variant of the vertex cover decision problem is NP-complete:

CONNECTED VERTEX COVER IN PLANAR GRAPHS WITH MAXIMUM DEGREE 4 DECISION PROBLEM
**Instance**: A planar graph $G = (V, E)$ with no vertex degree exceeding 4, and a positive integer $k$.
**Question**: Does there exist a vertex cover $C \subseteq V$ for $G$ satisfying $|C| \leq k$, such that the subgraph induced by $C$ is connected?

Note that this problem is more restricted than the classical vertex cover problem: the input graph is planar and each vertex has degree 4 or less. On the output side, we enforce a special requirement on the vertex cover — namely that the graph induced by $C$ is connected. More precisely, if we only keep the edges in $E$ that connect the vertices in the cover $C$, the resulting graph should be connected. (We can assume that $G$ is connected — otherwise the problem is trivial or can be reduced to the problem where $G$ is connected.)

**Theorem 3.19** *[170] The rectilinear Steiner tree decision problem is NP-complete.*

**Proof.** Let $G = (V, E)$ be a planar graph with no vertex degree exceeding 4, and let $k$ be a positive integer. We construct a terminal set $N$ in the plane for which a rectilinear Steiner tree $T$ of length at most $L$ exists *if and only if* $G$ has a connected vertex cover $C$ of size at most $k$. (The length bound $L$ is specified later.)

The first step in the construction of $N$ is to embed the planar graph $G = (V, E)$ in the plane. Let $n = |V|$ and $m = |E|$. Define the integer $\Delta = 2m + 2(k - 1) + 1$. Consider a grid of squares in the plane, where each grid point has integer coordinates that are multiples of $\Delta$. Map each vertex in $V$ to distinct grid points, and each edge in $E$ to non-intersecting grid paths (Figure 3.21). Such a *grid embedding* can be obtained in $O(n)$ time — and in such a way that the rectangular area covered has at most $O(n^2)$ grid squares [362].

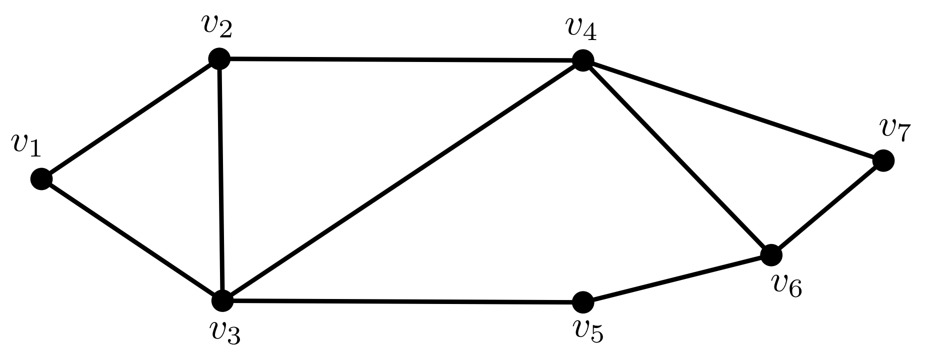
For each vertex $v_i \in V$, let $p_i$ be the corresponding grid point in the embedding. The terminal set $N$ is constructed by adding terminals located at each *integer coordinate* point lying on an edge in the embedding — *except* that no terminals are added within a distance of 2 from the points $p_i$ (Figure 3.22). An edge $e \in E$ that maps to a grid path of length $\kappa_e$ results in $\kappa_e - 3$ added terminals. Since the total length of all edges in the embedding is $O(\Delta n^2)$, the total number of terminals is $O(n^3)$, that is, polynomial in the size of $G$.

Imagine that we connect all terminals in $N$ that are distance 1 apart, as indicated by the red edges in Figure 3.22. These terminal-terminal connections are called *edge segments*. Each connected component of edge segments corresponds to an edge $e \in E$, and is called an *edge component*. The total length of all edge components is $\kappa = \sum_{e\in E}(\kappa_e - 4)$. The bound in the rectilinear Steiner tree decision problem is set to $L = \kappa + 2m + 2(k-1)$.

Now, assume that there exists a connected vertex cover $C \subseteq V$ for $G$ satisfying $|C| \leq k$. We show that there exists a rectilinear Steiner tree for $N$ of length at most $L$.

The rectilinear Steiner tree consists of all edge components plus a number of length 2 line segments that interconnect the edge components. Let $G_C = (C, E_C)$ be the connected subgraph that is induced by $C$ in $G$, and let $T_C = (C, E_T)$ be a tree spanning $C$ in $G_C$; note that $E_T$ contains $|C| - 1$ edges. (For example, if, for the graph in Figure 3.21, we take $C = \{\{v_2, v_3, v_4, v_6\}$, then a suitable choice of $E_T$ would be $E_T = \{(v_2, v_3), (v_3, v_4), (v_4, v_6)\}$.) For each edge $(v_i, v_j) \in E_T$, connect the corresponding edge component to the points $p_i$ and $p_j$ using two length 2 line segments. This results in a rectilinear Steiner tree that interconnects the grid points corresponding to the vertices in $C$. Finally, connect each remaining edge component to this tree using a single length 2 line segment. This is indeed possible since $C$ is a vertex cover (Figure 3.23). The total length of the rectilinear Steiner tree for $N$ is $\kappa + (2+2)(|C|-1) + 2(m - (|C|-1)) = \kappa + 2m + 2(|C|-1) \leq L$.

Conversely, let $T$ be a rectilinear Steiner tree for $N$ of length at most $L$. We can assume that $T$ is on the Hanan grid for $N$. Since the terminals in $N$ have integer coordinates, all line segments in $T$ have integer length. As a consequence, we can assume that $T$ contains all the edge segments of the edge components (each edge segment has length 1). To see why, suppose that one of these edge segments $uv$ is not in $T$. Adding $uv$ to $T$ creates a cycle that by the construction of $N$ contains at least one non-edge segment. By removing this non-edge segment, we obtain a tree of at most the same length that contains $uv$. This process can be iterated until all edge segments belong to $T$.



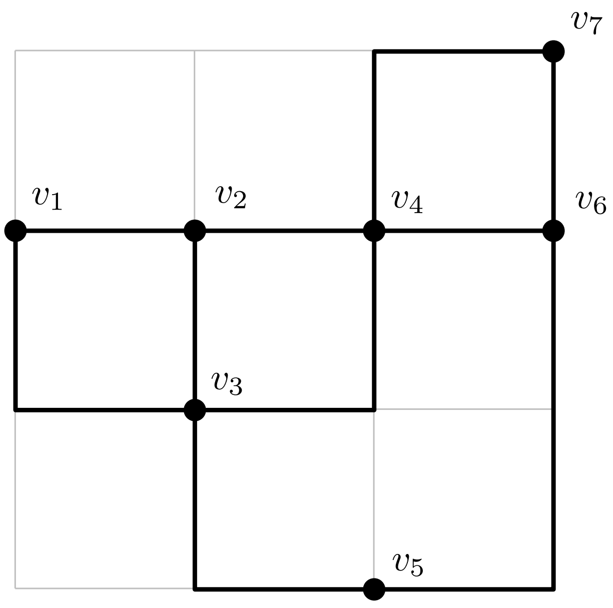


Figure 3.21: Embedding of planar graph (top) in a grid of squares (bottom). The vertex set $\{v_2, v_3, v_4, v_6\}$ forms a connected vertex cover of size 4.

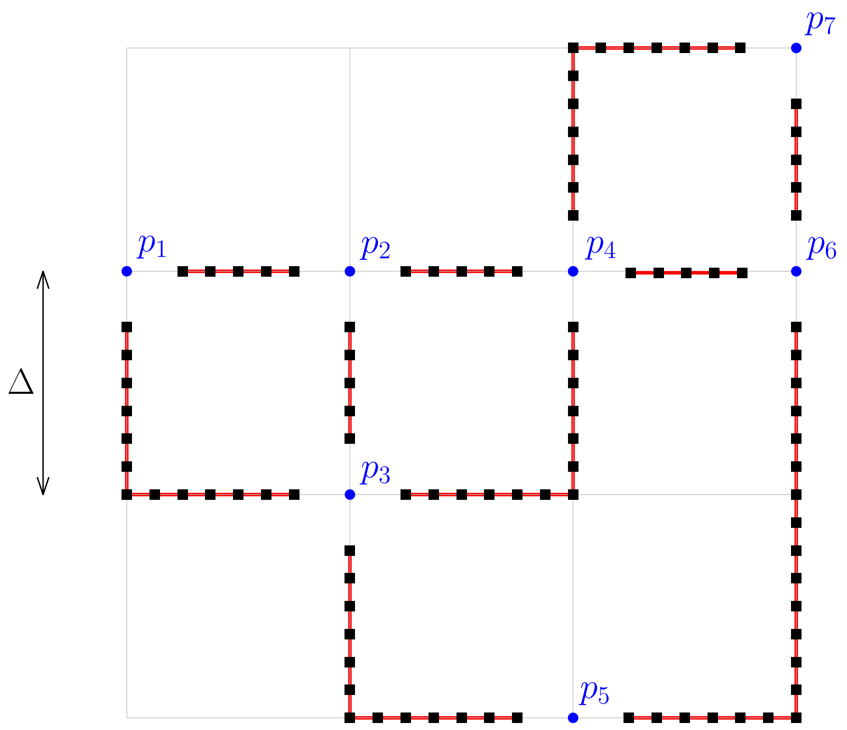


Figure 3.22: Construction of terminal set $N$. The terminals are indicated by the black squares. Each red connected component (i.e., edge component) corresponds to an edge $e \in E$.

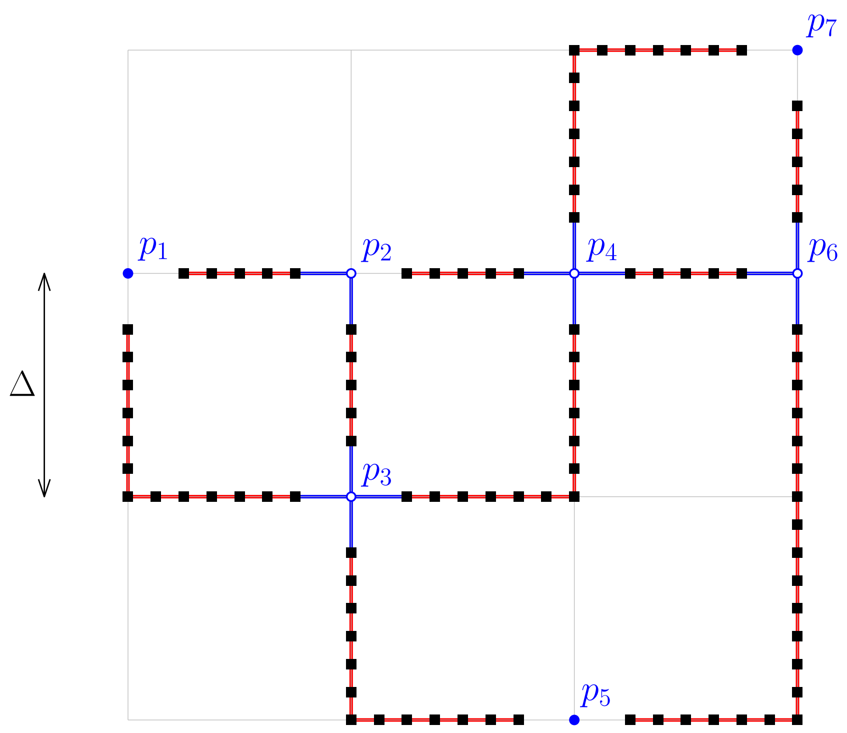


Figure 3.23: Construction of rectilinear Steiner tree for vertex cover $C = \{v_2, v_3, v_4, v_6\}$, where $E_T = \{(v_2, v_3), (v_3, v_4), (v_4, v_6)\}$. Length 2 line segments (blue) are added to interconnect the edge components (red).

So we may assume that $T$ contains all edge segments — or all terminal-terminal full Steiner trees of length 1 (the red edges in Figure 3.23). The remaining full Steiner trees of $T$ must therefore interconnect the edge components formed by the edge segments; that is, the terminals must come from different edge components.

**Claim.** Each of the remaining full Steiner trees of $T$ interconnects edge components that correspond to edges that meet at a *common* vertex in $G$.

**Proof of Claim.** Let $F$ be a full Steiner tree in $T$ that interconnects two terminals $u$ and $v$ that come from edge components that correspond to the edges $e$ and $f$ in $G$. Assume that $e$ and $f$ do not share an endpoint in $G$. In the grid embedding of $G$ this means that no points on the embeddings of $e$ and $f$ are within rectilinear distance $\Delta$ of each other. Thus the rectilinear distance between $u$ and $v$ is at least $\Delta$. Since terminals $u$ and $v$ are connected by a path in $F$, the length of this path is at least $\Delta$ — which implies that $F$ has length at least $\Delta$. As a consequence, $T$ has length at least $\kappa + \Delta = \kappa + 2m + 2(k-1) + 1 > L$ — a contradiction. ∎

Each of the remaining full Steiner trees in $T$ therefore interconnects two or more of the (at most four) edge components that meet at a single point $p_i$ in the embedding. Clearly, the shortest possible interconnection is either an L-shaped tree of length 4 that interconnects two edge components (see point $p_2$ in Figure 3.23), a T-shaped tree of length 6 that interconnects three edge components (see point $p_6$ in Figure 3.23) or a cross-shaped tree of length 8 that interconnects four edge components (see point $p_4$ in Figure 3.23). In every case we can think of each full Steiner tree as composed of length 2 line segments that join an edge component to a common point $p_i$.

Given $T$, we define the vertex cover of $G$ as follows:

$$C = \{v_i \in V : \text{some edge component is joined to } p_i\}.$$

Since $T$ is a tree that joins every edge component to some point $p_i$, $C$ is clearly a vertex cover. Also, since the edge components in $T$ have length $\kappa$, the remaining full Steiner trees have length at most $2m + 2(k-1)$. This means that at most $k-1$ out of the $m$ edge components are joined to two points $p_i$ and $p_j$ in $T$. Thus, if we delete all edge components that are joined to a *single* point $p_i$ in $T$, the resulting tree exactly interconnects the points corresponding to the vertex cover $C$, and contains at most $k-1$ edge components. Therefore, $C$ is connected in $G$ and $|C| \leq k$. ∎

Theorem 3.19 immediately implies the following corollary.

**Corollary 3.20** *The rectilinear Steiner tree problem is NP-hard.*

### 3.2.6 Equivalence to other problems with a pair of fixed orientations

As we have seen, the rectilinear metric is an example of a fixed orientation metric (as discussed in Chapter 2) where the two legal orientations are given by a pair of horizontal and vertical unit vectors. In this section we show that all of the results in this chapter can be directly generalised to any fixed orientation metric defined by exactly two linearly independent legal orientations, or equivalently to any Minkowski plane where the unit ball is a centrally symmetric quadrilateral.

The key result is the following theorem, showing an affine equivalence between the two problems.

**Theorem 3.21** *Given a pair of (weighted) legal orientations $\mathbf{u}_0, \mathbf{u}_1$ defining a fixed orientation metric, there exists an invertible affine transformation $A_{0,1}$ of the Euclidean plane with the following property: for any terminal set $N$, if $T_A$ is a minimum rectilinear Steiner tree for $N_A := A_{0,1}(N)$, then $T := A_{0,1}^{-1}(T_A)$ is a minimum fixed orientation Steiner tree for $N$ (for the given pair of orientations), and $\|T\| = |T_A|$.*

**Proof.** Given the set of fixed orientations $\{\mathbf{u}_0, \mathbf{u}_1\}$ and a terminal set $N$ with $n$ terminals, it follows from Theorem 2.29 (in Chapter 2) that there exists a minimum fixed orientation Steiner tree $T$ for $N$ where every Steiner point is in $\mathbf{GG}_{n-2}(N)$, the $(n-2)$th generalised Hanan grid for $N$. Since there are only two legal orientations it follows that $\mathbf{GG}_1(N) = \mathbf{GG}_2(N) = \cdots = \mathbf{GG}_{n-2}(N)$. Therefore, there exists a minimum fixed orientation Steiner tree $T$ in the Hanan grid graph $\mathbf{H}_{0,1}(N)$ for $N$: the vertices of $\mathbf{H}_{0,1}(N)$ are $\mathbf{GG}_1(N)$ (that is, the intersection points obtained by drawing lines in both legal orientations through every point in $N$) and the edges of $\mathbf{H}_{0,1}(N)$ are all line segments in legal directions connecting adjacent vertices; the edge weights in $\mathbf{H}_{0,1}(N)$ are the weighted Euclidean distances given by $\mathbf{u}_0$ and $\mathbf{u}_1$.

Let $o = (0,0)$ be the origin, and let $a_0$ and $a_1$ be the points such that $\overrightarrow{oa_0} = \mathbf{u}_0$ and $\overrightarrow{oa_1} = \mathbf{u}_1$. Furthermore, let $b_0$ and $b_1$ be the points with Cartesian coordinates $b_0 = (1,0)$ and $b_1 = (0,1)$. We define the transformation $A_{0,1}$ to be the invertible affine transformation that maps $\triangle oa_0a_1$ to $\triangle ob_0b_1$. Under this transformation the Hanan grid graph $\mathbf{H}_{0,1}(N)$ maps to the (rectilinear) Hanan grid graph $\mathbf{H}(N_A)$ for $N_A := A_{0,1}(N)$; this is illustrated in Figure 3.24. Note that not only do the two grid graphs have the same topology, but the length of each edge of $\mathbf{H}(N_A)$ is equal to the weighted length of the corresponding edge of $\mathbf{H}_{0,1}(N)$. It follows that solving the rectilinear Steiner tree problem for $N_A$ as a graph Steiner problem on $\mathbf{H}(N_A)$ immediately gives a solution to the fixed orientation Steiner tree problem on $N$ (by applying the inverse transformation $A_{0,1}^{-1}$) and that both trees have the same cost. ∎

Theorem 3.21 implies that solving any fixed orientation Steiner tree problem for a metric defined by two legal orientations is equivalent to solving the rectilinear Steiner tree

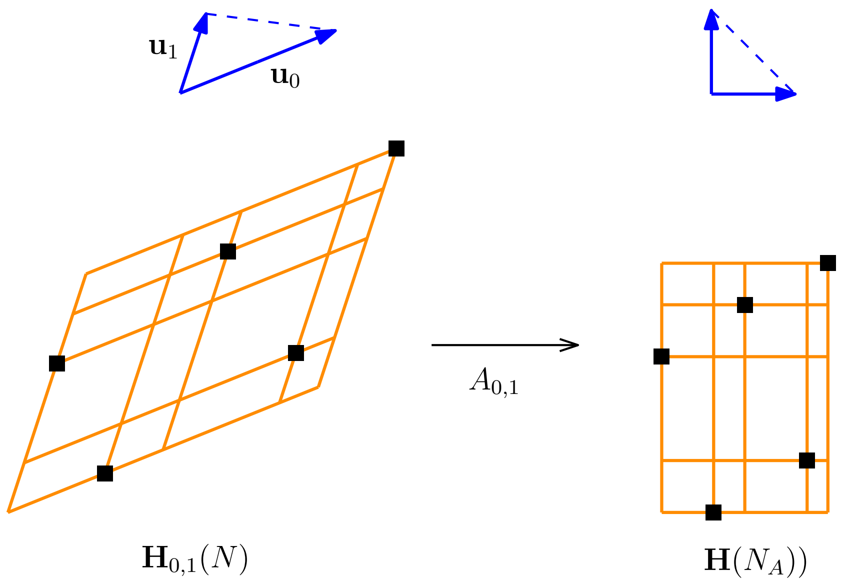


Figure 3.24: An example of the affine transformation $A_{0,1}$ for a set of five terminals. Here the fixed orientations $\mathbf{u}_0, \mathbf{u}_1$ are shown in blue and have weights $w_0 = 1/2$ and $w_1 = 1$, respectively.

problem. For example, this shows that the Steiner tree problem in the rectilinear metric is equivalent to the Steiner tree problem in the well-known $\ell_\infty$ (or maximum) metric, where the distance between two points is given by the maximum Euclidean distance between them in the horizontal and vertical directions.

In the proof of Theorem 3.21 the transformation is performed on the corresponding Hanan grid graph. In practice we only need to transform the terminal set (in linear time): compute and apply the affine transformation $A_{0,1}$ to the terminal set, solve the resulting rectilinear Steiner tree problem (using, for example, GeoSteiner), then apply the inverse transformation $A_{0,1}^{-1}$ to get the solution to the original problem. The final step of applying the inverse transformation can be omitted if only the length of the minimum Steiner tree is required.

Note also that there are numerous corollaries to Theorem 3.21 resulting from the properties of rectilinear Steiner trees discussed in this chapter. For example, it follows from this theorem and Theorem 3.10 that the Steiner ratio for any Minkowski plane where the unit ball is a centrally symmetric quadrilateral is $2/3$.

Finally, we mention that these results can be generalised, to a limited extent, to other Minkowski planes with polygonal unit balls. In particular, if the unit ball is an affinely regular polygon (meaning that it can be obtained from a regular polygon by an affine transformation), then an argument similar to the proof of Theorem 3.21 shows that the associated fixed orientation Steiner tree problem is equivalent to the uniform orientation Steiner tree problem for the corresponding regular polygon. Hence, in these cases the version of GeoSteiner for uniform orientation Steiner trees [297] can be used. Note, however, that for most arbitrary centrally symmetric convex polygons no such affine transformation exists.

## 3.3 GeoSteiner algorithm

The most efficient practical algorithms for computing minimum rectilinear Steiner trees are based on the GeoSteiner approach (see Section 1.4 in Chapter 1).[37] Recall that the main idea of the GeoSteiner approach is to enumerate full components — or full Steiner trees (FSTs) — and then choose a subset of the generated FSTs to form a minimum rectilinear Steiner tree. The first phase is called *FST generation* and the second *FST concatenation*. Salowe and Warme [335] presented the first serious attempt to apply the GeoSteiner approach to the rectilinear problem. A few years later, Zachariasen [429] and Warme [388] significantly improved the FST generation and FST concatenation parts of thr algorithm; a computational study reporting the solution of rectilinear Steiner tree problems with more than 1000 terminals was presented in [390].

Compared to other metrics, FST generation for the rectilinear problem appears to be particularly fast in practice. This is mainly due to the existence of the Hwang form for full components. Although the number of generated FSTs is exponential in the worst case (see Section 3.2.4), on average the number of generated FSTs is almost linear. In this section we first present the top-level FST generation algorithm. Then we describe some of the important details of the algorithm, including so-called *FST independent preprocessing* and the actual enumeration algorithm. As noted in Chapter 1, the FST concatenation phase of the algorithm is independent of the underlying metric, and can be reduced to either the Steiner tree problem in graphs (Section 5.1 in Chapter 5) or the minimum spanning tree problem in hypergraphs (Section 5.2.1 in Chapter 5).

### 3.3.1 Top-level FST generation algorithm

Consider a Hwang form FST $T$ in a minimum rectilinear Steiner tree, and assume that $T$ spans $k$ terminals $t_1, t_2, \ldots, t_k$. Recall that $T$ consists of a complete corner (or backbone) given by a root $t_1$ and a tip $t_k$. All other terminals spanned by $T$ are connected directly to the backbone with straight line segments, such that $T$ has a caterpillar topology.

Let $c$ be the corner point of the single bent edge $pq$ of $T$; if $T$ has no bent edge, then let $c$ be the midpoint of the straight edge $pq$ incident to the tip $t_k$ of $T$. Recall the definition of *branches* and *branch trees* from Section 1.4.2. If we cut edge $pq$ at $c$, we obtain two branch trees having *straight edges only*: one with its root at $p$ having a stem (or ray) leaving $p$ along $pc$, and another with its root at $q$ having a stem (or ray) leaving $q$ along $qc$. A *branch* is a set of branch trees that span a common set of terminals and that have a common topology.

In the construction below, it suffices to consider the generation of individual branch trees rather than branches. For a branch of size greater than $1$ with a given root and a given direction for the long leg we will see that under the construction scheme below the branch has a uniquely determined branch tree.

As in the Euclidean problem, we first describe how to combine branch trees to form larger ones. Branch trees of size 1 consist of a single terminal having a stem leaving in one of the legal directions; hence there are exactly $4n$ branch trees of size 1. The Hwang form for rectilinear FSTs and its caterpillar topology imply that we only need to consider combinations where one of the branch trees has size 1. More specifically, any full component $T$ can be obtained by starting with a branch tree $B_1$ consisting of the root of $T$ with a stem in the direction of the long leg. Then each of the terminals — in alternating fashion along the long leg — is iteratively added by combining the current branch tree with a size 1 branch tree that spans the added terminal. This results in a series of increasingly larger branch trees $B_1, B_2, \ldots$ as shown in Figure 3.25. A type (i) full component is obtained by combining a branch tree $B_i$ with a size 1 branch tree that spans the tip of the full component; finally, a type (ii) full component can be obtained by attaching a terminal to the short leg of the constructed type (i) full component (Figure 3.25, bottom row).

The simplified construction of branch trees for the rectilinear Steiner tree problem makes it possible to design a particularly efficient version of the general FST generation algorithm (Algorithm 1.3 from Section 1.4 in Chapter 1). Recall that one of the main tasks of the FST generation algorithm is to prune branches and full Steiner trees (lines 14 and 19 in Algorithm 1.3). Instead of enumerating branch trees by increasing size as in Algorithm 1.3, the main loop of the rectilinear FST generation algorithm iterates through all terminals $t_1 \in N$. For a given terminal $t_1$, all feasible FSTs that have $t_1$ as their *root* (in the Hwang form) are constructed. Each possible direction of the long leg is tried in turn; however, due to the existence of corner-flipped topologies (see Section 3.1.2), only two perpendicular directions need to be tried (say, north and east). The enumeration of feasible FSTs for a given (root, direction) pair is performed using a recursive algorithm that is described in Section 3.3.3. In the next section we first present a preprocessing step that significantly speeds up the enumeration algorithm.

### 3.3.2 FST independent preprocessing

In this section we describe an $O(n^2)$-time and -space preprocessing phase which is used to reduce the average complexity of the FST growing phase. (In fact, time and space

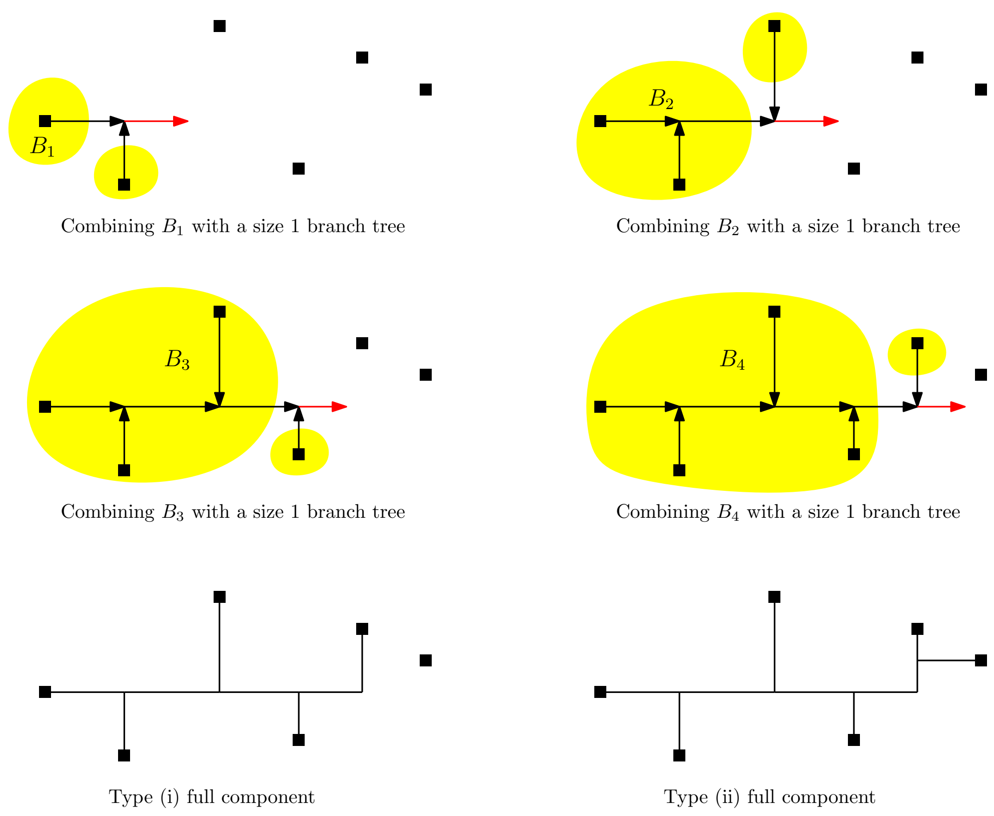

$B_1$
Combining $B_1$ with a size 1 branch tree
$B_2$
Combining $B_2$ with a size 1 branch tree
$B_3$
Combining $B_3$ with a size 1 branch tree
$B_4$
Combining $B_4$ with a size 1 branch tree
Type (i) full component
Type (ii) full component

Figure 3.25: Combining branch trees to form Hwang form full components of type (i) and type (ii). The arrows indicate the paths from the terminals to the root of each branch tree. The red arrow shows the direction of the new stem. The bottom row shows the resulting full type (i) and type (ii) full components.

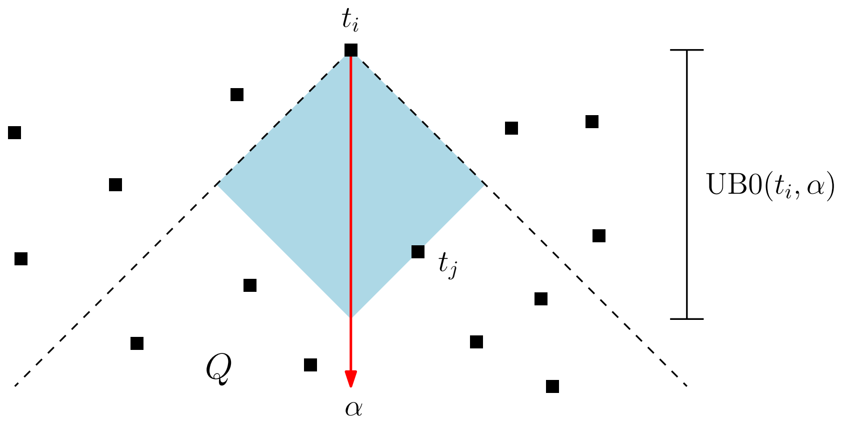


Figure 3.26: Upper bound based on lune property for inner terminal candidates.

requirements for this phase can be reduced to essentially $O(n \log n)$ in practice; see [429] for details.) First we set up the following data structures:

- Sort the terminals according to each of the four directions (north, south, east, west). For a given direction $\alpha$, let $N_\alpha(t_i)$ denote the successor of terminal $t_i$ in direction $\alpha$.
- Compute the bottleneck Steiner distance $\text{BSD}(t_i, t_j)$ for every pair of terminals $t_i, t_j \in N$; recall that the bottleneck distance bounds the length of each edge on a path between terminals $t_i$ and $t_j$ in a minimum rectilinear Steiner tree (Lemma 1.14 in Chapter 1).
- Determine whether or not the axis-aligned rectangle defined by $t_i$ and $t_j$, denoted by $\mathcal{R}(t_i, t_j)$, is free of terminals (for every pair of terminals $t_i, t_j \in N$).

### Inner terminal candidates

Let $(t_i, \alpha)$ be any (terminal, direction) pair. Consider a line segment $t_i s_i$ having direction $\alpha$ which attaches $t_i$ to some FST backbone via Steiner point $s_i$. We wish to compute an upper bound $\text{UB0}(t_i, \alpha)$ on the length of $t_i s_i$ such that if $t_i s_i$ is a part of some minimum rectilinear Steiner tree, then $|t_i s_i| \leq \text{UB0}(t_i, \alpha)$. We use the condition that the lune property must be fulfilled for $t_i s_i$, that is, $\mathcal{L}(t_i, s_i)$ must be free of terminals. Draw two perpendicular $45°$ lines through $t_i$ and let $Q$ be the quadrant in direction $\alpha$ given by these two lines (or, in other words, the cone from $t_i$ bounded by the two lines and containing the ray from $t_i$ in direction $\alpha$, as in Figure 3.26). From the lune property it follows that the distance from $t_i$ to the closest terminal $t_j$ in $Q$ is then a valid upper bound $\text{UB0}(t_i, \alpha)$. If there exists no terminal in $Q$, we set $\text{UB0}(t_i, \alpha) = \infty$.

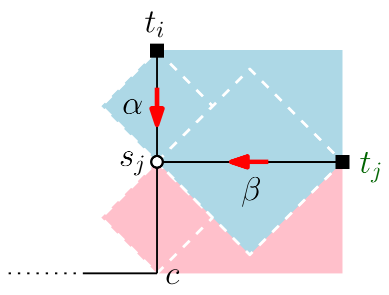


Figure 3.27: Determination of short leg terminal candidates and bounds for the corresponding corner points. For any candidate short leg terminal $t_j$ the regions shown in blue must be free of terminals. The maximum corresponding length of the short leg is then determined by the furthest potential corner point $c$ for which the pink regions are also free of terminals.

### Short leg terminal candidates

A short leg in a Hwang form FST has either zero or one attached terminal. Let $(t_i, \alpha)$ be any (terminal, direction) pair. Assume that $t_i$ is a tip in a Hwang form FST and that the short leg points in direction $\alpha$ from $t_i$. Since we only need to consider two perpendicular directions of the long leg when constructing Hwang form trees (say, north and east), a terminal can only be attached to one side of a short leg pointing in direction $\alpha$. We wish to determine an ordered list $N(t_i, \alpha)$ of terminals that may be attached to the left or right (depending on $\alpha$) of this short leg.

The key observations are the following. Let $t_js_j$ be a segment attached to the short leg of a backbone with tip $t_i$. Assume that the direction from $t_j$ to $s_j$ is $\beta$ (Figure 3.27). Then we must have:

- $|t_is_j| \leq \min(\text{UB0}(t_i, \alpha), \text{BSD}(t_i, t_j))$,
- $|t_js_j| \leq \min(\text{UB0}(t_j, \beta), \text{BSD}(t_i, t_j))$,
- $\mathcal{R}(t_i, t_j)$ is empty (contains no terminals).

The short leg candidates $N(t_i, \alpha)$ are identified by making a sweep from $t_i$ in direction $\alpha$. We use the first condition above to stop the scan when $|t_is_j| > \text{UB0}(t_i, \alpha)$. One implication of the third condition above is that the distance $|t_js_j|$ for any accepted terminal $t_j$ must be smaller than or equal to the shortest such distance seen during the sweep — otherwise the rectangle would be non-empty. The first two conditions are easily checked in constant time by using precomputed information.

Once $N(t_i, \alpha)$ is determined, we can compute an upper bound $\text{UB1}(t_i, \alpha)$ on the length of a short leg with one attached terminal with tip $t_i$ and pointing in direction $\alpha$.

Recall that UB0$(t_i, \alpha)$ already is an upper bound on the length of a short leg without any attached terminal.

If $N(t_i, \alpha) = \emptyset$, we set UB1$(t_i, \alpha) = 0$; this means that no short leg with an attached terminal exists. Otherwise, we seek the longest $t_i c$ such that there is a $t_l \in N(t_i, \alpha)$ having both an empty lune $\mathcal{L}(s_l, c)$ and an empty rectangle $\mathcal{R}(t_l, c)$. Let $t_j$ be the last terminal accepted into $N(t_i, \alpha)$, i.e., $|t_i s_j|$ is maximum. Then it is sufficient to check the empty regions for $t_j$ only, since for all other $t_l \in N(t_i, \alpha) \setminus \{t_j\}$ the regions $\mathcal{L}(s_j, c)$ and $\mathcal{R}(t_j, c)$ are covered by $\mathcal{L}(s_l, c)$ and $\mathcal{R}(t_l, c)$, respectively.

The upper bound UB1$(t_i, \alpha)$ is computed by making a sweep from $s_j$ (or equivalently $t_j$) in direction $\alpha$. The largest possible lune $\mathcal{L}(s_j, c)$ is identified by using the algorithm used for computing inner terminal candidates. The empty rectangle property is equivalent to testing whether there exists a terminal $t_l$ and corner point $c$ such that $|t_l c| < |t_j s_j|$; if such a terminal is encountered during the sweep, we set UB1$(t_i, \alpha) = |t_i c|$ and stop; otherwise, the value of UB1$(t_i, \alpha)$ is that given by the largest possible lune $\mathcal{L}(s_j, c)$.

## 3.3.3 Growing Hwang form full Steiner trees

The basic idea of the FST growing algorithm was presented in Section 3.3.1. In this section we give a more detailed description of the algorithm which uses the information obtained in the preprocessing phase. The main algorithm makes a call to the recursive algorithm (Algorithm 3.1) for every root $t_1 \in N$ and direction $\alpha \in \{\text{north}, \text{east}\}$, corresponding to a branch tree of size 1. With a slight abuse of notation, in the algorithm we view a branch tree $B$ as being both a branch tree and an FST candidate (by removing the stem of the branch tree).

The recursive algorithm performs four different types of tests, which are outlined below. Tests (I), (II) and (III) are particularly important as they eliminate the need for further recursive calls of the algorithm. The FST tests (IV) determine whether an FST candidate fulfils elementary optimality conditions. In the discussion below let $\beta$ be the direction from $t_i$ to the Steiner point $s_i$ on the long leg.

### Distance tests (I)

The upper bounds UB0 and UB1 obtained in the preprocessing phase are used to eliminate terminals from consideration. We must have $|t_i s_i| \leq \max(\text{UB0}(t_i, \beta), \text{UB1}(t_i, \beta))$ since otherwise the segment $t_i s_i$ can neither be attached to the long leg as an inner terminal nor be a short leg in an FST.

This test only depends on the root $t_1$ (in the Hwang form) and on the direction $\alpha$ of

**Algorithm 3.1:** Recursive FST growing algorithm

**Input**: Branch tree $B$ with root $t_1$ (in the Hwang form) and a direction $\alpha$. $B$ has segments $s_2t_2, \ldots, s_{i-1}t_{i-1}$ attached to the long leg.

**Output**: All Hwang form full Steiner trees that can be obtained by growing branch tree $B$ in direction $\alpha$.

```
// Determine candidate terminal successors for growing $B$
Let $N_\alpha(B)$ be the ordered set of terminals in direction $\alpha$, starting from $N_\alpha(t_{i-1})$
foreach $t_i \in N_\alpha(B)$ do
    // Assume that $t_i$ is attached to the long leg via Steiner point $s_i$
    if $t_i$ passes distance tests (I) then
        if $s_{i-1}s_i$ passes long leg segment tests (II) then
            $B_i = B \cup \{s_{i-1}s_i\} \cup \{t_is_i\}$ // new extended branch tree
            foreach short leg attachment candidate $t_j$ do
                // Assume that $t_j$ is attached to short leg $t_is_i$ via Steiner point $s_j$
                if $B_i \cup \{t_js_j\}$ passes type (ii) FST tests (IV) then
                    Save $B_i \cup \{t_js_j\}$ as a permanent type (ii) FST
            if $B_i$ passes branch tree tests (III) then
                if $B_i$ passes type (i) FST tests (IV) then
                    Save $B_i$ as a permanent type (i) FST
                Call algorithm recursively with input $B_i$ and direction $\alpha$
```

the long leg and not on the current branch tree $B$. This actually allows us to make a (short) list of candidates before calling the algorithm. However, since the FST growing algorithm typically stops well before reaching the end of the candidate list, we do not construct this list before calling the algorithm. Rather, we add candidates to the list during the execution of the recursive algorithm.

### Long leg segment tests (II)

This series of tests depends on the branch tree $B$ and the new long leg segment $s_{i-1}s_i$. A long leg segment $s_{i-1}s_i$ fails this test if the tree $B \cup \{s_{i-1}s_i\}$ cannot be a subtree of *any* larger FST that also spans $t_i$. We assume that $t_i$ is connected to $s_i$ either directly or via a short leg Steiner point.

Firstly, $t_i$ must be on the side of the long leg opposite to $t_{i-1}$. Secondly, the lune property must be satisfied for $s_{i-1}s_i$, and $\mathcal{R}(t_{i-1}, t_i)$ should contain no terminals. The lune property can be checked by maintaining an upper bound on the length of the segment

$s_{i-1}s_i$ based on the previous terminal candidates seen for $t_i$. Finally, the longest edge on the path between $s_i$ and any terminal $t_l$, $l \in \{1, \ldots, i-1\}$, in $B \cup \{s_{i-1}s_i\}$ cannot be longer than BSD$(t_i, t_l)$. This condition holds since the same (longest) edge will also appear on the path between $t_i$ and $t_l$ in any tree having $B \cup \{s_{i-1}s_i\}$ as a subtree. By updating this longest edge information dynamically while growing the tree this test can be performed very efficiently.

### Branch tree tests (III)

These tests check whether the branch tree $B_i = B \cup \{s_{i-1}s_i\} \cup \{t_i s_i\}$ can be a subtree of any larger branch tree or FST (including $B_i$ itself). We do this by testing the conditions $|t_i s_i| \leq$ UB0$(t_i, \beta)$ and $|t_i s_i| \leq$ BSD$(t_i, t_l)$ for all $t_l$, $l \in \{1, \ldots, i-1\}$.

### Type (i) and type (ii) FST tests (IV)

These tests check the optimality and fulsomeness of a specific FST candidate (of type (i) or type (ii)). One efficient test is the short leg upper bound given in Lemma 3.9, which can be tested in constant time by dynamically updating relevant information while growing the FST. The lune property (respectively, empty rectangle property) is also tested for every segment (respectively, pair of adjacent segments) both in the candidate topology and in the corner-flipped topology. (Many of these conditions are, in fact, satisfied by construction.) Furthermore, the empty inner rectangle property is checked (Lemma 3.15).

### Practical performance of the FST generation algorithm

For most problem instances the preprocessing phase dominates the total running time even though the recursive FST growing algorithm requires exponential time in the worst case. For randomly generated instances, the running time of the preprocessing phase grows as $O(n \log n)$, while the running time of the FST growing algorithm is close to being linear in $n$. The almost linear behaviour of the FST growing algorithm can be explained by the fact that very few terminals (less than 6 terminals for randomly generated problem instances with 10000 terminals) are added to the short list of terminals considered for a given root and direction [429].

On a modern computer, a well-tuned implementation of this algorithm generates the FSTs for a randomly generated 1000-terminal instance in less than 0.1 second; for 10000 terminals the running time is less than 2 seconds, and more than half of the time is used for preprocessing. The number of FSTs surviving all tests is approximately $4n$.

The bottleneck of the GeoSteiner algorithm for the rectilinear Steiner tree problem is therefore the concatenation of FSTs. As noted above, the FST concatenation problem can be reduced to either the Steiner tree problem in graphs or the minimum spanning tree problem in hypergraphs (see Chapter 5).

## 3.4 FLUTE algorithm

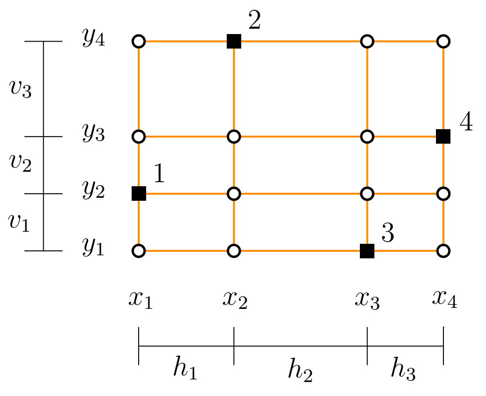
In the early stages of physical chip design, rectilinear Steiner trees are often used for estimating the necessary wire length in the final routing of the chip. By computing a minimum rectilinear Steiner tree for every electrical net, an accurate lower bound on the necessary wire length of the chip is obtained. In these applications, millions of rectilinear Steiner tree problems must be solved, so fast algorithms are required. Fortunately, most of the nets span very few terminals. For a typical chip, more than 90% of the nets span less than 10 terminals.

One extremely fast but rough wire length estimator is the *half-perimeter wire length (HPWL)* algorithm: find the smallest axis-aligned rectangle $R$ that contains the terminal set $N$, and compute the half-perimeter of $R$ (or the sum of the height and width of $R$). This estimate can clearly be computed in linear time by finding the minimum and maximum $x$- and $y$-coordinates amongst the terminals. HPWL is a lower bound on the length of a minimum rectilinear Steiner tree, and it gives the exact result for 2 and 3 terminals; for 4 or more terminals it usually underestimates the length of a minimum rectilinear Steiner tree.

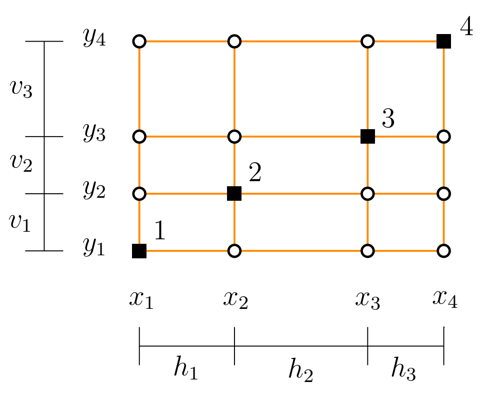
The FLUTE (fast lookup table estimation) algorithm [102, 103] is an exact algorithm for computing minimum rectilinear Steiner trees for small cardinality terminal sets. FLUTE is very fast for terminal sets with less than 10 terminals, and therefore useful as a wire length estimator in physical chip design. In this section we describe the main ideas of the FLUTE algorithm. We focus on how the *length* of a minimum rectilinear Steiner tree is computed by the FLUTE algorithm; a corresponding minimum rectilinear Steiner tree can be obtained by extending the lookup table [103]. It is convenient — but not necessary — to think of the terminals as being in general position, such that no two terminals are on the same grid line in the Hanan grid.

**Position sequence and wire length vectors**

Let $x_1 \leq x_2 \leq \cdots \leq x_n$ be the sorted sequence of $x$-coordinates for the terminals in the given terminal set $N$; similarly, let $y_1 \leq y_2 \leq \cdots \leq y_n$ be the sorted sequence of $y$-coordinates. Assume that the terminals are sorted by their $x$-coordinates such that the $i$th terminal, $1 \leq i \leq n$, has coordinates $(x_i, y_{\phi_i})$, and such that $\phi = (\phi_1, \phi_2, \ldots, \phi_n)$ is the corresponding permutation of the $y$-coordinates (see Figure 3.28). The permutation $\phi$ is called the *position sequence* for $N$.



Position sequence: (2,4,1,3)



Position sequence: (1,2,3,4)

Figure 3.28: Position sequences for two terminal sets.

In the Hanan grid for $N$, the $i$th vertical line has $x$-coordinate $x_i$, and the $j$th horizontal line (from the bottom) has $y$-coordinate $y_j$. Let $h_i := x_{i+1} - x_i$ be the distance between adjacent vertical grid lines for $1 \leq i \leq n-1$, and let $v_j := y_{j+1} - y_j$ be the distance between adjacent horizontal grid lines for $1 \leq j \leq n-1$ (see Figure 3.28). In the Hanan grid graph $\mathbf{H}(N)$ for $N$, all edge lengths clearly come from the two sets $\{h_1, h_2, \ldots, h_{n-1}\}$ and $\{v_1, v_2, \ldots, v_{n-1}\}$. Since there exists a minimum rectilinear Steiner tree for $N$ in $\mathbf{H}(N)$, the length of a minimum rectilinear Steiner tree $T$ can be expressed as

$$\sum_{i=1}^{n-1} \alpha_i h_i + \sum_{j=1}^{n-1} \beta_j v_j,$$

where the *integer* vector

$$W(T) := \{\alpha_1, \alpha_2, \ldots, \alpha_{n-1}, \beta_1, \beta_2, \ldots, \beta_{n-1}\}$$

is called the *wire length vector* for $T$ (Figure 3.29). Note that all $\alpha_i \geq 1$, and all $\beta_j \geq 1$, since otherwise the tree would not interconnect all terminals. A wire length vector that consists only of ones corresponds to the half-perimeter wire length (HPWL), and is called the *HPWL vector*.

The position sequence for $N$ uniquely places the $n$ terminals at specific grid points in the $n \times n$ Hanan grid — independent of the distances between adjacent horizontal lines and adjacent vertical lines. Therefore, for any two terminal sets $N$ and $N'$ with the same position sequence (and hence also the same number of terminals), any tree $T$ that spans $N$ in $\mathbf{H}(N)$ corresponds to a tree $T'$ that spans $N'$ in $\mathbf{H}(N')$. The trees $T$ and $T'$ have exactly the same topology relative to their respective Hanan grids; also, $T$ and $T'$ have the same wire length vectors.

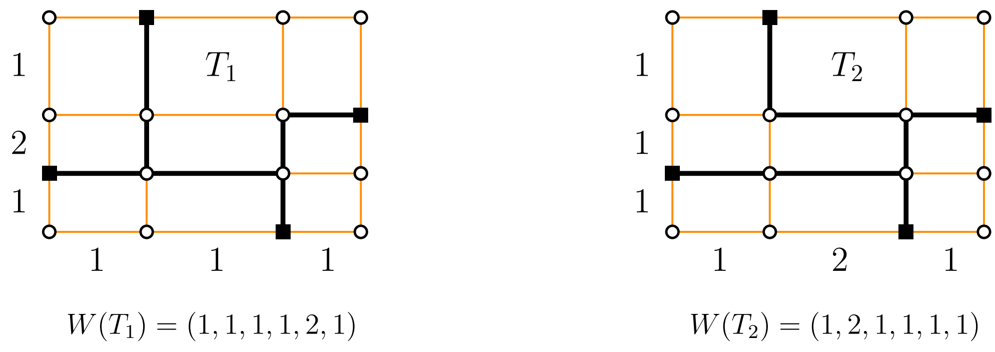


Figure 3.29: Two rectilinear trees and corresponding wire length vectors.

This observation leads to the main idea of the FLUTE algorithm. For a given position sequence $\phi$, imagine that we enumerate the set of trees $\mathcal{S}$ that span the terminals in the Hanan grid for a position sequence $\phi$ (ignoring the distances between adjacent horizonal lines and adjacent vertical lines). Each tree $T \in \mathcal{S}$ has an associated wire length vector $W(T)$.

**Definitions [Wire length domination, minimal wire length vector]**: For trees $T, T' \in \mathcal{S}$, we say that the wire length vector $W(T')$ *dominates* the wire length vector $W(T)$ if no component of $W(T')$ is greater than the corresponding component in $W(T)$, and at least one component of $W(T')$ is strictly smaller. A wire length vector $W(T)$ for a tree $T \in \mathcal{S}$ is a *minimal wire length vector* if there exists no other tree $T' \in \mathcal{S}$ such that $W(T')$ dominates $W(T)$.[38]

### Basic FLUTE algorithm

In order to compute the length of a minimum rectilinear Steiner tree for a set of terminals with position sequence $\phi$, we obviously only need to consider the set $\mathcal{W}_\phi$ of *minimal* wire length vectors for a position sequence $\phi$. For a wire length vector $W \in \mathcal{W}_\phi$, where $W = \{\alpha_1, \alpha_2, \ldots, \alpha_{n-1}, \beta_1, \beta_2, \ldots, \beta_{n-1}\}$, the length of a tree having wire length vector $W$ is denoted by $l(W) := \sum_{i=1}^{n-1} \alpha_i h_i + \sum_{j=1}^{n-1} \beta_j v_j$. The following lemma is immediate from the above discussion:

**Lemma 3.22** *Let $N$ be a terminal set with position sequence $\phi$, and let $\mathcal{W}_\phi$ be the set of minimal wire length vectors for $\phi$. Then the length of a minimum rectilinear Steiner tree for $N$ is*

$$\min_{W \in \mathcal{W}_\phi} l(W).$$

The HPWL vector $(1, 1, 1, 1)$ is the sole minimal wire length vector for *any* position sequence for 3 terminals (see Exercise 3.9). For 4 terminals, the position sequences are divided into two groups. The first group, which includes the position sequence shown in Figure 3.28 (right), has only one minimal wire length vector, the HPWL vector $(1, 1, 1, 1, 1, 1)$. The second group has two minimal wire length vectors, namely $(1, 1, 1, 1, 2, 1)$ and $(1, 2, 1, 1, 1, 1)$; see also Figure 3.29. Since the HPWL vector dominates any wire length vector, it always appears alone in a group.

The number of minimal wire length vectors for a given position sequence, that is, the worst-case cardinality of $\mathcal{W}_\phi$, appears to grow exponentially in $n$ (no asymptotic bounds are currently known). However, the absolute numbers are quite small for $n < 10$: for $n = 7$ the average and maximum numbers of minimal wire length vectors for a given position sequence are 8 and 15, respectively; for $n = 9$ the corresponding numbers are 30 and 79 [103]. This means that the minimum in Lemma 3.22 only needs to be taken over at most 79 minimal wire length vectors for $n < 10$, which is very fast in practice.

**Optimised FLUTE algorithm**

The basic FLUTE algorithm can be optimised in two ways. Firstly, we note that there are $n!$ position sequences for $n$ terminals. This rapid growth affects both the running time of minimal wire length vector enumeration, and the memory requirements, since all minimal wire length vectors for each position sequence must be stored.

One improvement is to use the symmetry of terminal sets, e.g., horizontal, vertical and diagonal symmetry. This makes it possible to transform minimal wire length vectors between symmetric position sequences. Another improvement is to partition the position sequences into groups that have the same minimal wire length vectors; as shown above, this reduces the lookup table for 4 terminals from $4! = 24$ to $2$ groups (or equivalence classes). In [103], these improvements reduce the number of groups by a factor of around 26 on average; the size of the lookup table is reduced by the same factor.

Secondly, the computation of the minimum in Lemma 3.22 can be optimised. We use the fact that minimal wire length vectors are very similar, and contain small integer numbers. Therefore, multiplications can be avoided when computing $\sum_{i=1}^{n-1} \alpha_i h_i + \sum_{j=1}^{n-1} \beta_j v_j$ by using additions and subtractions of numbers from the sets $\{h_1, h_2, \ldots, h_{n-1}\}$ and $\{v_1, v_2, \ldots, v_{n-1}\}$. Consider a position sequence for 4 terminals with associated minimal wire length vectors $(1, 1, 1, 1, 2, 1)$ and $(1, 2, 1, 1, 1, 1)$. We may compute the length of these two wire length vectors as follows. First we compute the HPWL $l$. Then we add $v_2$ to $l$ to obtain the length of the first wire length vector. Next we subtract $v_2$ and add $h_2$ to obtain the length of the second wire length vector.

More generally, in order to find a sequence of wire length vectors that minimises the total number of additions and subtractions, we can set up a complete graph with the wire length vectors, including the HPWL vector, as vertices. The distance between two vertices is the $\ell_1$ distance between their wire length vectors — which corresponds to the number of additions and subtractions needed to move from one vector to the other. A minimum-length Hamiltonian path, starting in the HPWL vector, now gives a wire length vector sequence that minimises the total number of additions and subtractions. In [103], the wire length vector sequence is obtained using a heuristic approach. For $n = 9$, given the HPWL, the length of a minimum rectilinear Steiner tree can on average be computed using 74 additions and subtractions, or around 2.5 additions/subtractions per wire length vector.

**Enumeration of minimal wire length vectors**

The FLUTE algorithm requires that the set of minimal wire length vectors $\mathcal{W}_\phi$ has been computed for every position sequence $\phi$. This is a preprocessing step that only needs to performed once, and therefore the running time is in principle not critical. However, in order to make it feasible to enumerate minimal wire length vectors for, say $n \geq 7$, it is necessary to devise a fairly efficient algorithm. The FLUTE enumeration algorithm is a recursive algorithm that involves a number of Hanan grid reductions, or *boundary compaction techniques*, that reduce the number of cases considered. For example, it can be assumed that $\alpha_1 = 1$, $\alpha_{n-1} = 1$, $\beta_1 = 1$ and $\beta_{n-1} = 1$ for any minimal wire length vector $\{\alpha_1, \alpha_2, \ldots, \alpha_{n-1}, \beta_1, \beta_2, \ldots, \beta_{n-1}\}$. The FLUTE enumeration algorithm is able to compute minimal wire length vectors for $n \leq 9$ [103].

## 3.5 Efficient constructions for special terminal sets

Although the rectilinear Steiner tree problem is NP-hard (as discussed in Section 3.2.5), there nevertheless exist exact algorithms, such as the GeoSteiner algorithm, that in practice appear to be able to solve the problem in low-order polynomial time for randomly and uniformly distributed terminal sets in the plane. To give some insight into this behaviour, we will survey some of the instances of configurations of terminals for which it can be shown that the rectilinear Steiner tree problem can be solved in polynomial time.[39] Many of these instances are also directly relevant to chip design applications. For example, the parallel lines cases, discussed below, where terminals are restricted to lying on a fixed set of horizontal (or vertical) lines, can be viewed as modelling the restriction of the routing problem where the pins are located on a fixed set of parallel tracks (see Section 3.6).

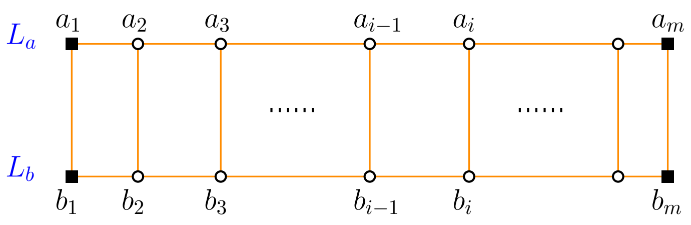


Figure 3.30: The Hanan grid graph for a set of terminals constrained to two horizontal lines. Here $a_1, b_1, a_m$ and $b_m$ are all terminals, and at least one of each vertical pair $a_i, b_i$ is a terminal.

### Terminals constrained to parallel lines

We first consider the case where all $n$ terminals $N$ lie on a pair of horizontal lines $L_a$ and $L_b$, and show that a minimum rectilinear Steiner tree for $N$ can be constructed in linear time. This provides an interesting contrast with the Euclidean case, where it has been shown that the Euclidean Steiner tree problem for terminals constrained to two parallel lines is NP-hard (see Section 1.3.3 in Chapter 1).

If the terminals $N$ are restricted to $L_a$ and $L_b$ then the Hanan grid graph $\mathbf{H}(N)$ is a $2 \times m$ grid, for some $m < n$, as illustrated in Figure 3.30. Here the vertices of $\mathbf{H}(N)$ are $a_1, \ldots, a_m, b_1, \ldots, b_m$ where the $a_i$'s and $b_i$'s lie on $L_a$ and $L_b$ respectively. We can assume that the four nodes $a_1, b_1, a_m$ and $b_m$ are all terminals, since otherwise the reduced Hanan grid graph has terminal leaves which can be removed; see Section 3.2.2. Moreover, for each pair $a_i, b_i$ of vertices of $\mathbf{H}(N)$ at least one of the vertices is a terminal in $N$.

This Hanan grid graph $\mathbf{H}(N)$ is an example of what is known as a series-parallel graph, meaning it contains no subgraph homeomorphic to $K_4$. It follows from a result of Wald and Colbourn [383] on Steiner trees on graphs (see Chapter 5) that Steiner trees on such graphs can be constructed in linear time. Here, however, we outline a more direct constructive proof based on an earlier approach by Aho, Garey and Hwang [4].

Let $T$ be a minimum rectilinear Steiner tree for $N$. For each positive integer $i \leq m$ let $\mathbf{H}_i(N)$ be the induced subgraph of $\mathbf{H}(N)$ with vertices $a_1, \ldots, a_i, b_1, \ldots, b_i$ and let $T_i$ be the part of $T$ contained in $\mathbf{H}(N)$ (in other words, $T_i$ consists of the parts of $T$ on and to the left of $a_i b_i$). With this definition of $T_i$ we have the following lemma.

**Lemma 3.23** *For a terminal set $N$ constrained to two horizontal lines, there exists a minimum rectilinear Steiner tree $T$ such that each subgraph $T_i$ is a tree.*

This lemma can be proved by sliding all vertical components of $T$ as far to the left as possible without disconnecting the tree and its terminals (Exercise 3.10).

Let $N_i$ be the set of all terminals in $a_1, \ldots, a_i, b_1, \ldots, b_i$. Lemma 3.23 suggests that it should be possible to construct $T$ from left to right by, for example, a simple dynamic

**Algorithm 3.2:** Minimum rectilinear Steiner tree on two parallel lines algorithm

```
Input: Set of terminals $N$ lying on two horizontal lines $L_a$ and $L_b$.
Output: A minimum rectilinear Steiner tree $T$ for $N$.

Let $\mathbf{H}(N)$ be the Hanan grid graph for $N$, with vertex set $\{a_1, \ldots, a_m, b_1, \ldots, b_m\}$
  where the $a_i$'s and $b_i$'s lie on $L_a$ and $L_b$ respectively
Set $T_a(1) = T_b(1) = T_{ab}(1) = a_1 b_1$
for $i = 2$ to $m$ do
    Let $T_{ab}(i) =$ the minimum length tree from $\{ T_a(i-1) \cup \{a_{i-1}a_i\} \cup \{a_i b_i\},$
                                                  $T_b(i-1) \cup \{b_{i-1}b_i\} \cup \{a_i b_i\},$
                                                  $T_{ab}(i-1) \cup \{a_{i-1}a_i\} \cup \{b_{i-1}b_i\} \}$
    if $b_i \in N$ then
        Let $T_a(i) = T_{ab}(i)$
    else
        Let $T_a(i) = T_a(i-1) \cup \{a_{i-1}a_i\}$
    if $a_i \in N$ then
        Let $T_b(i) = T_{ab}(i)$
    else
        Let $T_b(i) = T_b(i-1) \cup \{b_{i-1}b_i\}$
    Compute the lengths $|T_{ab}(i)|, |T_a(i)|, |T_b(i)|$
$T = T_{ab}(m)$
```

programming approach. An algorithm along these lines is given in Algorithm 3.2. As $i$ increases from 1 to $m$, the algorithm recursively keeps track of the three possible (not necessarily disjoint) forms that the subtree $T_i$ can take:

$T_{ab}(i) =$ a minimum rectilinear Steiner tree on $N_i \cup \{a_i, b_i\}$;

$T_a(i) =$ a minimum rectilinear Steiner tree on $N_i \cup \{a_i\}$;

$T_b(i) =$ a minimum rectilinear Steiner tree on $N_i \cup \{b_i\}$.

It is straightforward to prove the correctness of this algorithm; see Exercise 3.11.

Clearly each of the constructions in lines 5 to 13 of the algorithm can be performed in constant time, and the updating of the lengths of each of the candidates for $T_i$ (in line 14) can then also be done in constant time based on these constructions. This results in the desired linearity result.

**Theorem 3.24** *[4] A minimum rectilinear Steiner tree for a terminal set constrained to two horizontal lines can be constructed in linear time.*

The dynamic programming strategy underlying Algorithm 3.2 can be extended to the construction of a minimum rectilinear Steiner tree $T$ for terminals $N$ constrained to any fixed number $k$ of horizontal lines. Again the tree can be built up from left to right by constructing candidate subgraphs for each $T_i$, the part of $T$ that lies on or to the left of a vertical line through vertices in the Hanan grid. One of the main differences in this more general case is that $T_i$ is no longer necessarily a tree, but in general will be a Steiner forest with multiple components. The difficulty of keeping track of all possible combinations of components and all possible optimal updates of $T_i$ as $i$ increases means that the complexity of constructing $T$ is exponential in $k$, the number of horizontal lines, but is still linear in $n$.

A general algorithm of this type was developed by Brazil, Thomas and Weng [67], leading to the following result.

**Theorem 3.25** *[67] A minimum rectilinear Steiner tree for a set of $n$ terminals constrained to $k$ horizontal lines can be constructed by an algorithm with time complexity $O(nk^3 10^k)$ and space complexity $O(nk5^k)$.*

### Terminals on rectilinearly convex polygons

Recall, from Section 3.2.2, that a *path-convex hull* for a set of terminals $N$ in the rectilinear plane is defined to be a rectilinear polygonal region containing all elements of $N$ such that a walk around its perimeter has minimum possible length. A key property of path-convex hulls is that there always exists a minimum Steiner tree for $N$ contained in any path-convex hull of $N$. The boundary of a path-convex hull is sometimes referred to as a *rectilinearly convex boundary*, and the hull itself as a *rectilinearly convex polygon*. We will make use of this terminology in the survey that follows.

Here we consider the case where all terminals $N$ lie on the boundary of a rectilinearly convex polygon $R_k$ with $k$ sides. Note that we can assume that each side of the polygon contains at least one terminal, since otherwise we could reduce it to a problem with a smaller $k$. This implies that $R_k$ is a path-convex hull of $N$.

One of the earliest approaches for developing a polynomial-time algorithm for the case where all terminals lie on the boundary of a path-convex hull of $N$ was proposed by Provan [321]. Provan observed that in this case solving the rectilinear Steiner tree problem is equivalent to solving the Steiner tree problem in graphs (see Chapter 5) where the underlying graph is the reduced Hanan grid graph for $N$ (as illustrated, for example in Figure 3.14) and where all terminals lie on the outer boundary of the graph. Note also that any leaf edges of the reduced Hanan grid graph (such as those incident with $t_1, t_2, t_5$ and $t_7$ in Figure 3.14) can be collapsed down to zero-length edges to give an equivalent problem where the reduced Hanan grid graph is a 2-*connected* planar graph (meaning that it is connected, and remains connected if any single edge is deleted), and all terminals lie on the boundary of the exterior face. Such a planar graph is referred to as $1$-*planar*.

The Steiner tree problem in graphs for 1-planar graphs has been shown (via a straightforward dynamic programming algorithm) to be solvable in polynomial time. In particular, a result of Bern [31] on $1$-planar graphs translates into an $O(n^5)$-time algorithm for solving the corresponding Steiner tree problem. This was later improved to an $O(n^3)$-time algorithm by Cheng et al. [90].

**Theorem 3.26** *[90] Let $N$ be a set of $n$ terminals that lie on the boundary of the reduced Hanan grid for $N$. Then a minimum rectilinear Steiner tree for $N$ can be constructed by an algorithm with time complexity $O(n^3)$.*

If we return to the case of finding a minimum rectilinear Steiner tree $T$ for a set of $n$ terminals on the boundary of a given rectilinearly convex polygon $R_k$ with $k$ sides, then stronger results are possible, particularly if $k$ is significantly smaller than $n$. An algorithm for solving this problem with time complexity $O(nk^4)$ was presented by Richards and Salowe in [326]. The key to developing such an algorithm is to show that there are only a limited number of possible locations for any full component of $T$ containing at least one Steiner point.

To take a representative case, suppose there is a full component $T_0$ of $T$ that has Hwang form of type (ii) (as defined in Theorem 3.8). This is illustrated in Figure 3.31. We define the *extended backbone* of $T_0$ to be the union of the complete corner of $T_0$ with the segment incident to the short leg and with the segment incident to the long leg closest to the corner point. For example, the extended backbone of the type (ii) full component in Figure 3.31 is indicated in bold. For each corner point of $R_k$ with interior angle of $3\pi/2$, Richards and Salowe identify at most 10 associated grid lines of the Hanan grid which they define to be *blue lines*, and then they show that the extended backbone of $T_0$ is blue (that is, is composed only of segments of blue lines). Since there are $O(k)$ blue lines there are $O(k^4)$ possibilities for the extended backbone of $T_0$.

Each possibility for the extended backbone of $T_0$ divides $R_k$ into four subregions (indicated by the four different coloured regions in Figure 3.31) and hence decomposes the problem of constructing $T$ into four subproblems. Richards and Salowe [326] show that each of these subproblems can be solved in time $O(k^4+n)$, using a dynamic programming approach based on solving the subproblems for regions of increasing area. This results in an $O(k^8 + k^4n)$-time algorithm, which they show, through detailed case analysis, can be improved to $O(k^4n)$. More recently, Cheng and Tang in [91] and [92] have shown that the running time can be further improved to $O(k^2n)$ by refining the dynamic programming method used for the subproblems.

**Theorem 3.27** *[91] Let $R$ be a rectilinearly convex polygon with $k$ sides and let $N$ be a*

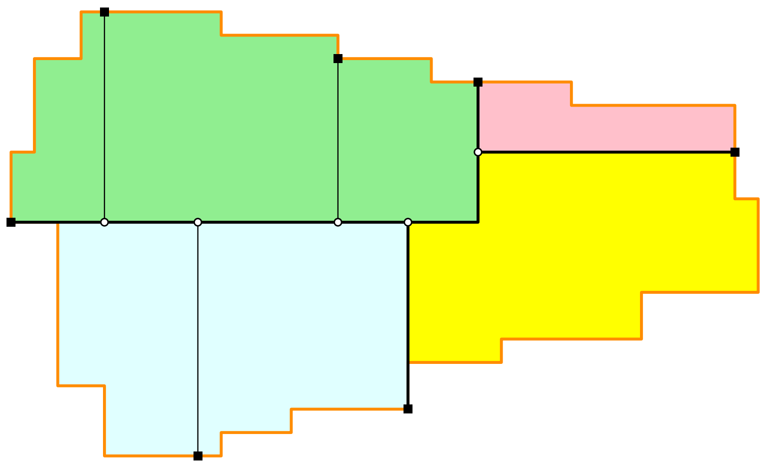


Figure 3.31: A full component (shown in black) for terminals on a rectilinearly convex boundary (shown in orange). Only terminals belonging to this full component are shown. The extended backbone of the full component (indicated by the bold black lines) partitions the rectilinearly convex polygon into four subregions (indicated by the different colours).

*set of $n$ terminals that lie on the boundary of $R$. Then a minimum rectilinear Steiner tree for $N$ can be constructed by an algorithm with time complexity $O(k^2n)$.*

We note that Cheng [89] has also developed a similar algorithm for terminals lying on the boundary of a non-convex rectilinear polygon, showing that a minimum rectilinear Steiner tree $T$ can be constructed in time $O(k^3n)$ as long as $T$ lies inside the polygon.

## Terminals constrained to curves

We conclude this section with a brief discussion of a more general class of instances that admit polynomial-time solutions, namely terminals constrained to lying on a given set of compact simple curves. The result is analogous to Theorem 1.21 for the Euclidean Steiner tree problem in Chapter 1. As with that theorem, the result here is more of theoretical than practical interest, as the degree of the polynomial may be high (depending on the geometry of the given set of curves). The theorem, however, gives some insight into the boundary between P and NP for the rectilinear Steiner tree problem.

Before stating the theorem, we require some simple definitions. For a given compact curve $C$ in the plane, we define a *rectilinear segment* of $C$ to be a maximal closed straight line segment which is either vertical or horizontal. Furthermore, a non-differentiable point $p$ in the interior of $C$ is said to be an *abnormal point* if there exists a supporting line for $C$ at $p$ which is either vertical or horizontal.

**Theorem 3.28** *[64] Let $G$ be a finite union of disjoint compact simple continuous curves in the plane that are smooth almost everywhere and contain a finite number of abnormal points and rectilinear segments. Then, there is a polynomial-time algorithm for solving the rectilinear Steiner tree problem for any set of terminals lying on $G$.*

Note that in the above theorem we treat the set $G$ as a given quantity. Here, polynomial time refers to time polynomial only in $n$, the number of terminals. The degree of this polynomial, however, is highly dependent on the nature of $G$.

The overall strategy for proving Theorem 3.28 is similar to that used in the proof of Theorem 1.21. For the given set $G$ we define a finite set of *capsules* covering $G$; each capsule is a region consisting of a small neighbourhood around a section of one of the curves in $G$. The capsules are defined in such a way as to guarantee that there are only a polynomial set of possibilities for the topology of the part of a minimum rectilinear Steiner tree $T$ within the capsule (in terms of the number of terminals in the capsule). It can also be shown that there are only a bounded number of edges leaving each capsule, giving a finite bound on the number of possible topologies for the part of $T$ outside the capsules.

One of the main challenges in the proof is to find a polynomial-time algorithm for constructing any part of $T$ in a capsule containing an abnormal point, as these capsules can contain an unbounded number of Steiner points. The proof here makes use of a result of Bern and Bienstock [29] that gives a polynomial-time solution for the Steiner tree problem on planar graphs where the terminals all lie within a bounded number of layers of the infinite face. For the full proof of Theorem 3.28 see Brazil et al. [64].

## 3.6 Applications and extensions

The rectilinear Steiner tree problem plays an important role in the physical realisation of electronic circuits. The physical design of circuits is about mapping a circuit onto an almost planar surface — or converting a circuit description into a geometric description.

The construction of an electronic circuit begins by describing the behaviour of the circuit. The process of specifying the logic functions of the circuit and their interrelations, often referred to as *logic synthesis*, is performed using a hardware description language (HDL). Logic synthesis produces a *netlist*, which describes how a set of standard components such as NANDs or NORs are interconnected. For each component, a specific physical realisation is chosen, depending on requirements related to area consumption, load capacitance and timing. A component with a specific physical realisation is called a *module*, and it can be considered to be a rectangle with given dimensions. Each *net* of the netlist should interconnect a set of *pins* from different modules. One of the pins is the *source*, that is, the pin that drives the electric signal of the net.

In its simplest form, the physical design of a circuit is about placing the modules of the circuit without overlap onto a given planar surface, and connecting the pins of each net such that wires from different nets do not intersect. In this section we first describe some of the steps involved in this mapping procedure — and in particular those where minimum rectilinear Steiner trees play a role. Then we describe some of the extensions of the rectilinear Steiner tree problem that are motivated by the physical design of circuits. Finally, in this section, we briefly discuss extensions of the rectilinear Steiner tree problem to higher dimensions.

### 3.6.1 Physical design of circuits

The modules of a *printed circuit board* consist of individually packed electronic components (e.g., resistors, capacitors or more complex integrated circuits) that are placed on a non-conductive planar board. The wires are 'printed' on the surface of the board by removing parts of a copper-layer using using etching or milling techniques (Figure 3.32); the planar board can consist of several layers, allowing wiring on multiple layers. The orientation of wires on a single layer is typically rectilinear or octilinear (see also Section 2.7 in Chapter 2). The history of printed circuit boards goes back to the early twentieth century. Before the invention of printed circuit boards, components were connected using point-to-point wire connections. Printed circuit boards reduced the cost of circuits, and significantly increased robustness and reliability.[40]

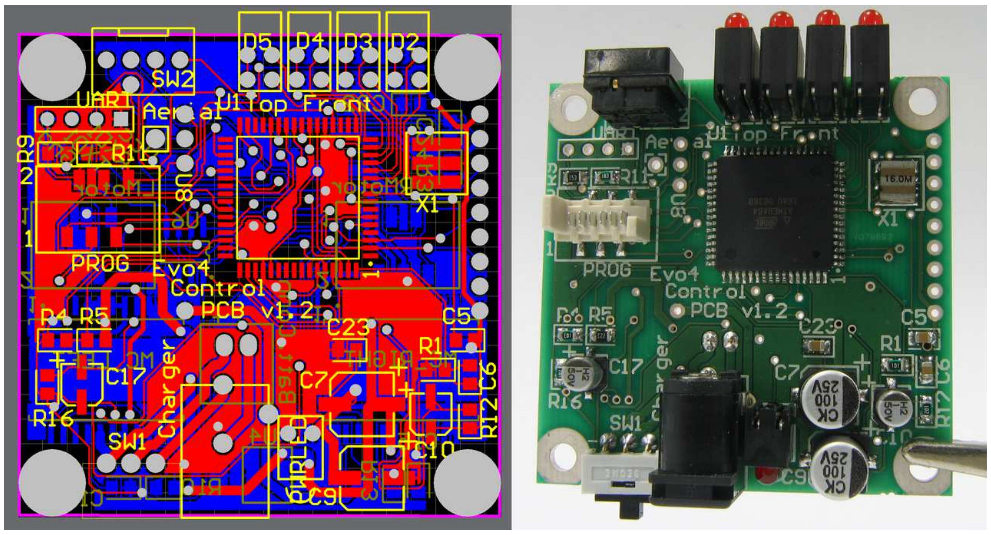


Figure 3.32: Printed circuit board design (left) and realisation (right). In the design, the placement of the modules (yellow), the wire layout on the front side (red) and the wire layout on the back side (blue) are shown.

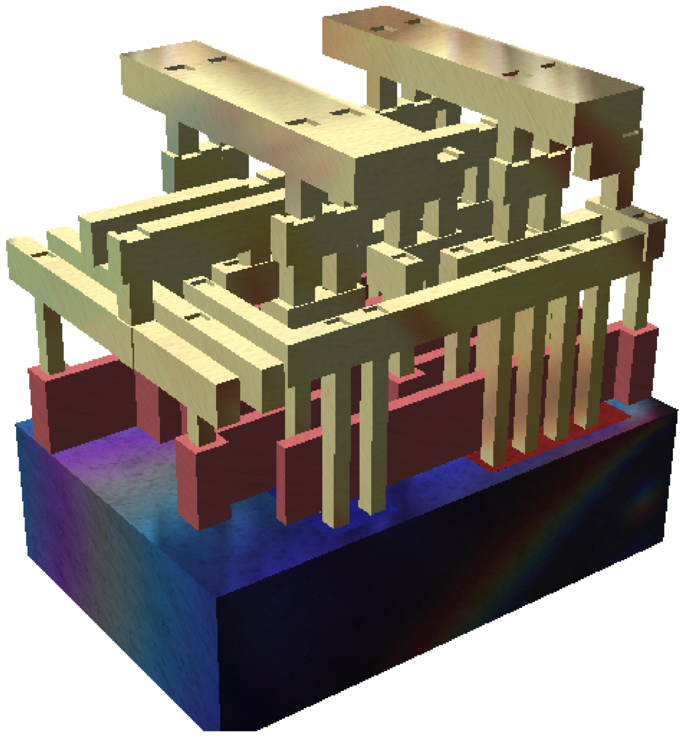


Figure 3.33: Model of a small integrated circuit with three metal layers (insulator has been removed). The sand-coloured structures at the top are lines of metal interconnect. The layers are connected using vias (vertical pillars). The reddish middle structures are polysilicon gates, and the solid layer at the bottom is the substrate.

An *integrated circuit* (also referred to as an IC or a chip) is a set of electronic circuits on one small plate (chip) of semiconductor material. Integrated circuits are fabricated on silicon wafers (also called the substrate). By marking different areas of the substrate using photolithography, patterns/tracks consisting of polysilicon, insulator or metal can be deposited on the substrate (Figure 3.33). In this way transistors and the wires connecting them can be built on a very small scale on the surface of the substrate — a process often referred to as very-large-scale integration (VLSI) design. Current technology makes it possible to construct patterns/tracks less than 25 nanometres wide. For an introduction to algorithms for the physical design of integrated circuits, see Alpert et al. [11] and Held et al. [194].[41]

In this section the focus is on integrated circuits, rather than printed circuit boards. In printed circuit boards, almost all nets are simple paths between pairs of pins, and hence the theory of Steiner trees is less relevant to the physical design. Here one of the principal optimisation challenges in the physical design stage is efficiently packing all the nets on a small number of layers [1].

### Main steps of physical design: placement and routing

The overall objective of the physical design of printed circuit boards and integrated circuits is to obtain a feasible placement of the modules, and a feasible routing of the wires. Also, as a general rule, the wire length of individual nets, as well as the total wire length of all nets, should be minimised. Short interconnections minimise area usage and reduce the signal delay of nets. In the following, we describe the steps of physical design for integrated circuits (or chips). The steps can be iterated in order to improve the performance of the chip.

The first step of physical design is usually *floorplanning*, where major parts of the circuit are placed on the chip surface. (For an integrated circuit of a CPU, such major parts could be arithmetic logic units, branch predictors, caches etc.) In the *placement* step each of the modules is located on the chip surface. The placement problem is a multi-objective problem, where area usage, wire length and signal delay are the primary objectives. These objectives are usually combined into a single quality measure called *netlength*. The problem of meeting timing (or clock rate) constraints is called *timing optimisation*. Timing can be improved by inserting amplifiers on wires (buffer insertion) and by adjusting the size of transistors (gate sizing). Also, timing can be optimised by adjusting the netlength of critical nets and reoptimising the placement under the new netlength objective.

The final step of physical design is *routing*, where the wires interconnecting the modules are located on the chip surface and assigned to different (metal) layers. Wires on a given layer have a preferred direction which is either horizontal or vertical — the so-called *Manhattan routing*. Each net of the netlist should interconnect a given set of pins on the chip surface, and wires from different nets should not intersect each other. The main objectives of routing are firstly to obtain a physically feasible routing layout and secondly to minimise the total wire length and signal delay. Figure 3.34 illustrates a small chip where placement and routing has been performed.

### Global and detailed routing

Routing is performed with the help of a (flat) three-dimensional grid graph, where the distance between neighbouring grid lines is the minimum width of a wire plus the minimum distance between wires. In older technologies modules and wires were aligned perfectly on the grid graph, but in current designs, modules and wires can be placed off-grid; such routing is usually called *gridless* or *shape-based* routing. Furthermore, wires are typically wider on the upper metal layers of the chip, allowing timing-critical long-distance nets to

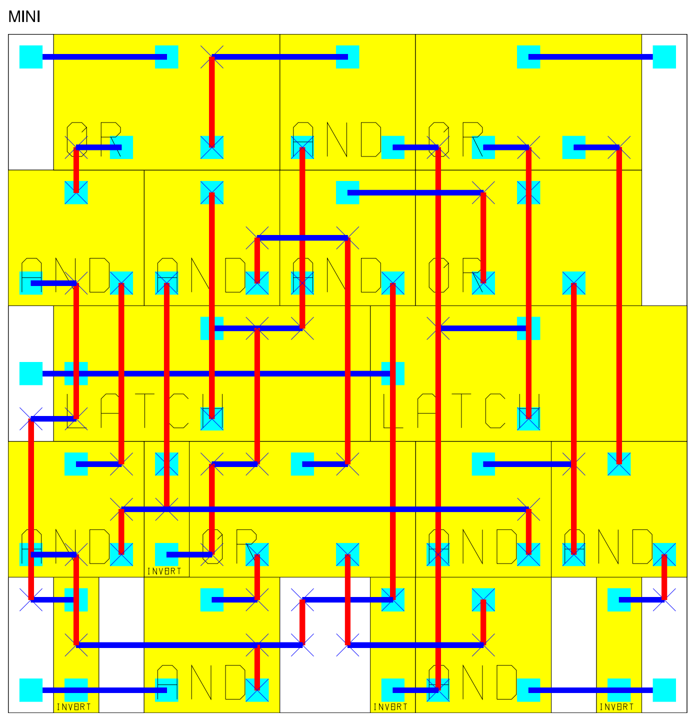


Figure 3.34: A placement and a routing of a small circuit. The circuit has 19 modules, 22 nets and 58 pins. The horizontal blue wires and vertical red wires run on different metal layers. Vias are indicated by $\times$'s. (Figure reproduced by courtesy of Research Institute for Discrete Mathematics, University of Bonn.)

be routed there. Modern routing algorithms therefore only implicitly use the grid graph.

Routing millions of nets in a grid graph with billions of nodes is a challenging task. The problem is therefore divided into at least two steps: *global routing* and *detailed routing*. The global routing problem is a coarse version of the routing problem, where the chip surface is divided into axis-aliged rectangular regions. The height and width of a region is typically 50–100 grid lines in the grid graph. In the *global routing grid graph* the vertices are the regions, and two vertices are connected by an edge if the two regions are neighbours. The edges in the global routing grid graph have associated lengths and capacities, where the capacities estimate the maximum number of wires that can be routed between two neighbouring regions. The global routing problem is, in its simplest form, a so-called Steiner tree packing problem, where trees should be 'packed' in the global routing grid graph such that the capacities of the edges are respected.

The output of the global routing problem is a 'global routing corridor' for each net, that is, a coarse description of the wiring of each net. In the detailed routing problem the exact wiring of each net is determined — and in such a way that the output of global routing is respected. Using the output from global routing both minimises the risk of congestion in detailed routing and speeds up detailed routing, since only a relatively small part of the full grid graph needs to be considered. Due to the size of the detailed routing problem, the problem is normally solved one net at a time — and in most cases one point-to-point connection at a time (however, state-of-the-art routing software can perform these routing tasks in parallel for a single chip). While global routing is a multi-objective problem where congestion, timing and wire length are considered, detailed routing is primarily concerned with feasibility, and the only real optimisation involved is (implicit) shortest-path computation in the grid graph.

#### Technological constraints: design rules

Over the last 50 years, chip producers have managed to halve the size of transistors (and width of wires) every 2 years or so — a development often referred to as Moore's law. For example, Intel has moved from a 130 nanometre technology to a 14 nanometre technology over a time span of 12 years (from 2002 to 2014). Due to the shrinking scales of wires, chip production processes and the physics related to small scales enforce still more so-called *design rules* that must be met by the final routing. Such rules not only involve distances between different nets (*diff-net rules*), but increasingly also involve rules prohibiting certain configurations of shapes within a single net (*same-net rules*). Examples include constraints on the minimum length of wire segments, and constraints on the minimum area of a single wire on a layer. Such design rules make it increasingly difficult just to use simple optimisation algorithms, and the problem can only be solved partially through postprocessing of a proposed routing solution.

#### The role of rectilinear Steiner trees in chip design

Although every net on a modern chip is an almost planar rectilinear Steiner tree, the direct use of minimum rectilinear Steiner trees in chip design has its limitations. Firstly, a routed net is not planar, and the cost of connecting wires on different layers cannot be ignored. Secondly, a net cannot be routed freely on the surface of the chip, but has to avoid all other nets of the chip. And thirdly, design rules and optimisation criteria other than wire length may require rectilinear Steiner trees that are not necessarily length-minimal. In the next section we discuss some of the extensions of the rectilinear Steiner tree problem that are motivated by the physical design of circuits.

### 3.6.2 Extensions motivated by the physical design of circuits

In this section we present a number of extensions of the rectilinear Steiner tree problem that are motivated by chip design. The presentation is not comprehensive and does not cover all the literature, but it does cover many of the most important extensions.

### Wire length estimation

One of the important direct applications of the rectilinear Steiner tree problem is wire length estimation (see also Section 3.4). A minimum rectilinear Steiner tree provides a very accurate *lower bound* on the necessary wire length in the placement phase of physical chip design. In this phase, complicating constraints can be — or rather must be — ignored; therefore, computing a minimum rectilinear Steiner tree for every net provides a good indication of the quality of a placement solution.

More generally, wire length can be estimated using a number of different *netlength models*. The purpose of a netlength model is to quantify the quality of a given placement; minimising total wire length is usually a primary objective, but other objectives such as signal delay and power consumption also play an important role. A minimum rectilinear Steiner tree obviously has minimum length, but is NP-hard to compute. Therefore, computational methods that can rapidly estimate the minimum length — and that can be incorporated directly and efficiently into the objective function of the placement algorithm — are of great interest.

In addition to minimum rectilinear Steiner tree length and half-perimeter wire length estimates (see Section 3.4), a number of other methods can be used to estimate wire length. Consider a set $N$ of $n$ terminals. Brenner and Vygen [75] compare the following alternative estimates: the length of a minimum rectilinear spanning tree for $N$ [324], the star length of $N$ (i.e., a solution to the general Fermat problem for $N$) and clique length (i.e., the sum of distances over all pairs of points in $N$ divided by some function of $n$). Brenner and Vygen [75] argue that the clique model is most appropriate in placement algorithms that only consider two-terminal connections, that is, where a fixed topology must be assumed for each net of the chip.

### Delay-driven routing

In the routing problem the task is to interconnect the pins/terminals of every net of the chip. For each net $N$, one of the terminals $r \in N$ is the *source*, while the remaining terminals in $N$ are the *sinks*. The electrical signal should propagate from the source to the sinks via the constructed tree. Thus, the rectilinear Steiner tree is in fact directed — or a so-called *Steiner arborescence*.

One of the important objectives that should be taken into account is the *signal delay* from the source to the sinks. In particular, if the net is part of the *critical signal path* of the chip, then the signal delay of the constructed tree has a direct influence on the clock rate (or performance) of the chip. Signal delay is related to the lengths of the paths from the source to the sinks, so minimising total path length often improves the signal delay properties of the tree. However, the actual signal delay has more complex behaviour, and depends not only on the length of the path itself, but also on the lengths of the subtrees that are attached to the path. Furthermore, for a simple two-terminal connection, signal delay increases quadratically with the length of the connection. The popular Elmore delay model [19, 149, 204, 231, 305, 344] serves as a good estimation for computing the signal delay. The slightly simpler distributed RC delay model is easier to use for optimisation, but serves best as an upper bound on the delay [120]. An overview of models and techniques for optimising delay can be found in [40, 118, 231]. The problem of minimising Elmore delay is still a major algorithmic challenge in chip design [240].

Other simpler models can be used to incorporate delay into rectilinear Steiner tree algorithms. The simplest model is to assume that the delay of a wire is linear in its length. Ignoring all other objectives, a shortest-path tree would provide minimum delay. The problem of constructing a rectilinear tree of minimum length in which every source-sink path is a shortest rectilinear path is called the *rectilinear Steiner arborescence problem* [323]. Unfortunately, shortest-path trees usually have unacceptable high total length. Therefore, so-called *shallow-light* trees have attracted considerable interest [9, 20, 119, 234, 292]. These are trees that have short paths from their source to their sinks (are ‘shallow’) and also have small total length (are ‘light’). Other relatively simple models that incorporate delay have been proposed by Bartoschek et al. [24] and Held and Rotter [195].

For most nets on the chip, a minimum rectilinear Steiner tree has satisfactory delay properties — assuming buffers have been inserted to meet timing constraints. Furthermore, a minimum rectilinear Steiner tree is usually not unique, and different embeddings may have different delay properties. Bozorgzadeh et al. [41, 42] discussed methods that can exploit the flexibility of embeddings of rectilinear Steiner trees. Boese et al. [33, 34, 35] introduced a *Global Slack Removal* algorithm that attempts to improve the delay properties of a rectilinear Steiner tree without increasing its length. Peyer et al. [307] took this idea one step further and considered the problem of constructing a minimum rectilinear Steiner tree with some *secondary* delay-related objective. More specifically, they focussed on the problem of constructing a minimum rectilinear Steiner tree with the weighted sum of path lengths as the secondary objective (i.e., path lengths from a given source to a set of sinks). An optimal solution to this problem exists in the Hanan grid for the terminal set [430], but Peyer et al. [307] proved the following stronger result:

**Theorem 3.29** *In a rectilinear Steiner tree problem equipped with a secondary objective of minimising a weighted sum of the path lengths, the Steiner points of an optimal solution* must *overlap with vertices of the Hanan grid for the terminal set.*

Adding the secondary objective to the problem thus forces the Steiner points to belong to the Hanan grid. Peyer et al. [307] presented both exact and heuristic algorithms for the problem. The heuristic algorithm, denoted *Extended Global Slack Removal (XGSR),* is also capable of minimising Elmore delay as a secondary objective. Experiments with real-life chip instances with 4 to 40 terminals were presented. (Note that minimum rectilinear Steiner trees with 2 or 3 terminals are always optimal with respect to the weighted sum of path lengths.) On average, XGSR constructed secondary objective optimal trees for 98.4% of the problem instances; only 52.0% of the problem instances were optimal before applying XGSR.

**Group interconnections**

During the global routing phase of chip design, it is usually assumed that each pin of the net is a single point. Thus, computing a minimum length interconnection is the same as computing a minimum rectilinear Steiner tree for the pins of the net. However, on a real chip a pin typically consists of several rectangles (or line segments), and any point on this set of rectangles suffices as a connection point. This fact motivates the study of so-called *group* Steiner trees, where each 'terminal' consists of a set of rectangles. The roots of this problem go back to Melzak [281], who discussed the Euclidean group problem (where the groups are convex sets in the plane).

As shown by Zachariasen and Rohe [434], the rectilinear group problem with rectangles reduces to the rectilinear group problem with points, as the problem can be solved in the Hanan grid of the corner points of the given rectangles. So from here on we consider the following definition of the rectilinear group Steiner tree problem: Given a set of groups $\mathcal{N} = \{N_1, N_2, \ldots, N_k\}$, where each group $N_i$, $i = 1, \ldots, k$, is a finite set of points in the plane, construct a minimum rectilinear Steiner tree which spans at least one point from each group; such a tree is called a *minimum rectilinear group Steiner tree.* This problem is NP-hard even for very restricted cases, e.g., when all the terminals are required to lie on two parallel lines [217] or when Steiner points are not allowed [218]. The group problem can be transformed to the ordinary Steiner tree problem in a graph by introducing so-called super-terminals. Therefore, the problem can be solved using any exact algorithm for the Steiner tree problem in graphs (see Chapter 5).

Zachariasen and Rohe [434] gave a first (tailored) exact algorithm for solving the rectilinear group Steiner tree problem. They presented techniques to reduce the given set of points, that is, to remove points in the groups $N_i$ from consideration by showing that an optimal tree exists that does not use these points. Also, a generalised version of Zachariasen's [429] full Steiner tree generation algorithm was used to reduce the Hanan grid graph — hence speeding up standard branch-and-cut algorithms for solving the corresponding graph problem. Computational experiments on real-life and random problem instances with up to 100 groups were performed. The techniques employed resulted in a speed-up approaching an order of magnitude, and increased the range of practically solvable real-life problem instances from around 40 groups to beyond 70 groups.

### Obstacle-avoiding interconnections: hard and soft obstacles

When solving the routing problem in chip design, certain regions of the chip surface may be forbidden — or may have certain restrictions. Such regions are usually denoted *obstacles*. Obstacles typically consist of pre-placed macros or other circuits (Figure 3.35). In older technologies, where the number of available layers was limited, routing across pre-placed circuits was impossible. These circuits formed *hard* obstacles. In newer technologies, where the number of layers is higher, it is possible to route wires across pre-placed circuits; however, routing across such *soft* obstacles is usually less attractive, since these regions have less interconnect capacity and/or no space for placing buffers/repeaters.

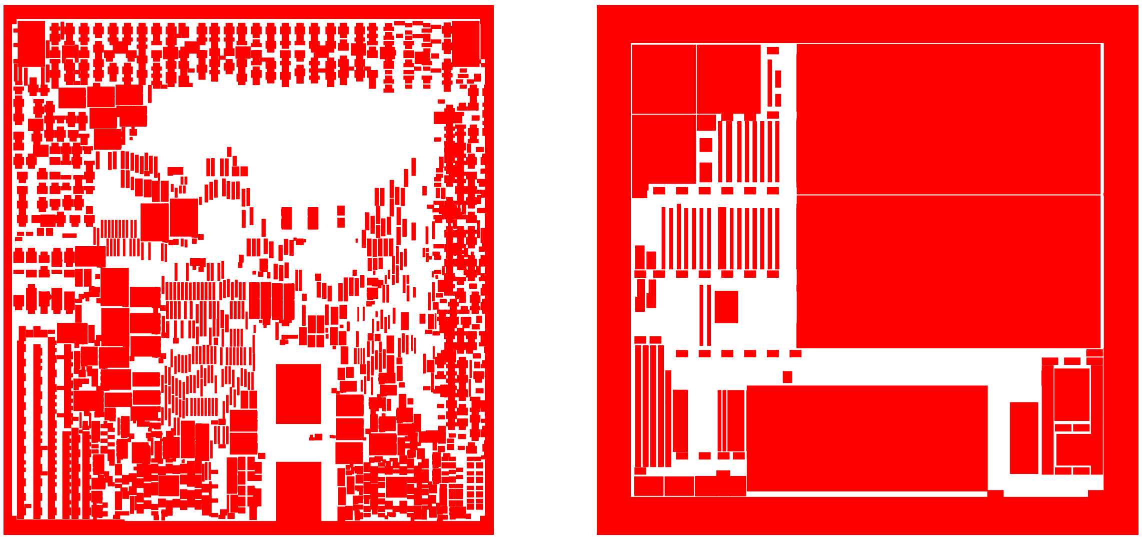


Figure 3.35: Typical distribution of obstacles on two real-life chips. (Pictures reproduced by courtesy of Research Institute for Discrete Mathematics, University of Bonn.)

Of these two routing problems, the problem of constructing a minimum length tree that interconnects a given set of terminals and avoids a set of *hard* polygonally bounded obstacles has by far received the most attention in the literature. Ganley and Cohoon [164] observed that the obstacle-avoiding rectilinear Steiner tree problem can be solved in a subset of the Hanan grid of the terminals and obstacle corners — thus providing an efficient reduction to the Steiner tree problem in planar graphs. The obstacle-avoiding problem for hard obstacles is studied in depth in Section 4.2 of Chapter 4.

For the rectilinear Steiner tree problem with *soft* obstacles, a number of different variants may be considered:

**Uniform weight obstacles.** Each obstacle has a given weight, and the problem is to find a minimum rectilinear Steiner tree, where the length of any part of the tree that crosses an obstacle is multiplied by the weight. The regions correspond to congestion hot-spot regions where previous placement or routing iterations identified potentially high levels of congestion. There exists an optimal solution to this problem in the Hanan grid of the terminals and obstacle corners [430], so this problem reduces to the Steiner tree problem in planar graphs. The problem of computing shortest rectilinear paths with weighted obstacles was studied in [245, 248].

**Non-uniform weight obstacles.** This variant is similar to the previous variant except that the weight of an obstacle is not uniform, but decreases closer to the boundary. The idea is that there is a central region (a congestion hot spot) that has a high weight and an outer region where the weight decreases as one moves away from the central region (see also [158]).

**Length-bounded intersections.** In this variant the problem is to find a minimum rectilinear Steiner tree, such that each maximal component of the tree that overlaps with the interior of an obstacle has length bounded by some constant. An interesting special case is the problem where all terminals are on the boundary of a single rectilinear obstacle. Approximation algorithms for the length-bounded intersection variant have been proposed in [196, 290].

**Slew-constrained intersections.** This variant is similar to the previous variant, but here the constraint on the overlap depends on the delay properties of the overlapping tree. An exact algorithm for this variant is given in [209].

#### Other extensions

Here we briefly mention some other extensions of the rectilinear Steiner tree problem that have been studied.

**Floating Steiner trees.** Sarrafzadeh et al. [339] studied rectilinear *floating* Steiner trees. In this problem multiple Steiner trees share a terminal that is movable. This problem is relevant in the placement phase of chip design. The problem is to find the position of the terminal that minimises overall tree length.

**Minimum segment lengths.** In this problem the task is to find a minimum rectilinear Steiner tree with the restriction that there is a given *lower bound* for the length of any segment — due to lithographic constraints in the production of the chip. More generally, there may be a collection of similar lithographic constraints that the tree must satisfy [278, 296].

### 3.6.3 Extensions to higher dimensions

As described in the previous sections, the three-dimensional rectilinear Steiner problem has obvious applications in the physical design of integrated circuits (since such circuits are in fact three-dimensional). In this section we briefly present some of the main properties of rectilinear Steiner trees in $d$-dimensional space (with a focus on $d \geq 3$), and cover some interesting applications of higher dimensional rectilinear Steiner trees (see also [213]).

For any two points $p = (x_1, x_2, \ldots, x_d)$ and $q = (y_1, y_2, \ldots, y_d)$ in a $d$-dimensional space, their $\ell_1$ distance is $|pq|_1 = |x_1 - y_1| + |x_2 - y_2| + \cdots + |x_d - y_d|$, that is, the sum of distances in each of the $d$ dimensions. Formally, the $d$-dimensional rectilinear Steiner tree problem is as follows:

RECTILINEAR STEINER TREE PROBLEM IN $d$-DIMENSIONAL SPACE
**Given**: A set of points $N = \{t_1, \ldots, t_n\}$ lying in $d$-dimensional space (for $d \geq 2$).
**Find**: A geometric network $T = (V(T), E(T))$, such that $N \subseteq V(T)$, and such that $|T| := \sum_{e \in E(T)} |e|_1$ is minimised.

As in the planar case, a network solving the rectilinear Steiner tree problem in $d$-dimensional space is a tree, referred to as a *minimum rectilinear Steiner tree*. All the other associated definitions including terminals, Steiner points, and full components carry across to this more general setting in a natural way.

## Generalisation of the Hanan grid property

The rectilinear Steiner tree problem in $d$-dimensional space can be reduced to a Steiner tree problem on a grid graph. More precisely, Snyder [356] generalised the Hanan grid property (see Section 3.2.2) to higher dimensional spaces:

**Theorem 3.30** *[356]* ***[Hanan grid reduction in $d$-dimensional space]*** *Let $N$ be a set of terminals in $d$-dimensional space (for $d \geq 2$). There exists a minimum rectilinear Steiner tree for $N$ such that every Steiner point is located at a point whose coordinates appear in $N$.*

It follows that the rectilinear Steiner tree problem in $d$-dimensional space can be solved on a grid graph with $O(n^d)$ vertices and $O(dn^d)$ edges, where $n = |N|$; the size of the grid graph is polynomial in $n$ for constant dimension $d$. Du and Hwang [137] generalised Snyder's result to any $d$-dimensional normed space with a unit sphere that is a centrally symmetric polytope with $2d$ extreme points. (Du and Hwang's proof is much simpler than Snyder's original proof.)

## Non-existence of Hwang forms for dimensions $d \geq 3$

The existence of the Hwang form for full components has played a pivotal role in the design of exact algorithms for the rectilinear Steiner tree problem in the plane (see Section 3.3). One of the key features of the Hwang form for a full component is that it is a *caterpillar* tree, meaning that the subtree induced by the Steiner points is a single path. Wulff-Nilsen [412] has shown that for $d \geq 3$ we can no longer assume that each full component is a caterpillar. For a given full component $T$ and a Steiner point $s$ in $T$, let the *Steiner depth* of $s$ be the minimum distance (measured in the number of edges) to a terminal in $T$. The Steiner depth of $T$ is then the *maximum* Steiner depth for any Steiner point in $T$. The Steiner depth can be at most $O(\log n)$ for any full component that spans $n$ terminals (Exercise 3.12). A full component which is a caterpillar clearly has Steiner depth 1.

**Theorem 3.31** *[412]* ***[Full components for $d \geq 3$ can have large Steiner depth]*** *For any dimension $d \geq 3$ and any $k \in \mathbb{N}$ there exists a set of terminals $N$ of cardinality $n = 2^k + 1$, such that there is a* unique *minimum rectilinear Steiner tree $T$ for $N$, and such that $T$ is a single full component with Steiner depth $\Theta(\log n)$.*

This theorem leaves little hope of finding simple characterisations of full components in minimum rectilinear Steiner trees for $d \geq 3$, and suggests that any algorithm for solving the rectilinear Steiner tree problem in $d$-dimensional space will have to consider all possible Steiner topologies for full components.

## Exact algorithms

A basic approach for computing a minimum rectilinear Steiner tree is to enumerate all full Steiner topologies (where Steiner vertices have degree 3 and terminals have degree 1), and to compute a relatively minimal tree for each topology (see Section 1.1.3 in Chapter 1). Sankoff and Rousseau [337] gave a dynamic programming algorithm for locating the Steiner vertices for a given topology. Here we sketch the Sankoff-Rousseau algorithm for the case where the topology is a full Steiner topology (see also [412]).

The main observation is that we may consider each of the $d$ dimensions separately, since optimising the rectilinear length for one dimension does not affect the rectilinear length for any other dimension. Let $\mathcal{T}$ be a full Steiner topology for terminal set $N$. Set the root of the topology to be one of the terminals $r \in N$. The idea is now to process the Steiner vertices bottom-up in the rooted (binary) tree/topology as in the Melzak-Hwang algorithm for the Euclidean Steiner tree problem in the plane (see Section 1.2.1 in Chapter 1). For a given dimension and Steiner vertex $s$, we identify a simple interval in which Steiner vertex $s$ must be located in order to obtain a minimum rectilinear Steiner tree for the subtree rooted at $s$. The interval can be computed in constant time given the corresponding intervals for the two children of $s$ in $\mathcal{T}$.

When the bottom-up algorithm reaches the root, we do an inverse top-down traversal fixing the locations of each Steiner vertex — also in constant time per vertex for a given dimension. The construction results in a unique canonical minimum rectilinear Steiner tree for the given topology and root $r$ [337, 412].

**Theorem 3.32** *[337]* ***[Sankoff-Rousseau algorithm]*** *Let $N$ be a set of $n$ terminals in $d$-dimensional space (for $d \geq 2$), and let $\mathcal{T}$ be a full Steiner tree topology for $N$. Then we can compute a relatively minimal rectilinear Steiner tree for $N$ and $\mathcal{T}$ in $O(dn)$ time.*

The relatively minimal tree that is obtained by the Sankoff-Rousseau algorithm has the property that the coordinates of the Steiner vertices only come from the given terminals, and therefore the existence of the canonical tree immediately implies Theorem 3.30.

It seems likely from Theorem 3.31 that any GeoSteiner-type algorithm must take all possible full Steiner topologies into consideration when generating full Steiner trees (FSTs) for $d \geq 3$. Wulff-Nilsen, in [412], has implemented a GeoSteiner algorithm for the $d$-dimensional rectilinear Steiner tree problem. Despite the application of several sophisticated FST-pruning tests, the algorithm was able to compute minimum rectilinear Steiner trees for only up to around 15 terminals for $d = 3$ and around 10 terminals for $d = 4, 5, 6$. Wulff-Nilsen [412, 413] has also given some bounds on the expected number of FSTs in higher dimensions fulfilling certain necessary conditions.

For some classes of problem instances the number of different coordinates for each dimension may be low — leading to a small number of Steiner point candidates in the grid graph. Chowdhury et al. [101] exploited this fact in the design of a spanning tree enumeration algorithm (see Section 5.1 in Chapter 5) for the rectilinear Steiner tree problem in higher dimensions.

### Relation to phylogenetic trees, Wagner trees and Hamming distance problems

Rectilinear Steiner trees in higher dimensions appear to have been mentioned first in the context of the construction of *phylogenetic* trees in the 1960s. A *phylogeny* is a tree representing the ancestral relationship for a set of species. A species is represented by a sequence $\alpha_1\alpha_2\ldots\alpha_d$ of characters (e.g., nucleotides or proteins), and the difference, or distance, between two species $\alpha_1\alpha_2\ldots\alpha_d$ and $\beta_1\beta_2\ldots\beta_d$ is measured as the number of sites $i \in \{1,\ldots,d\}$ where $\alpha_i \neq \beta_i$ — also called the *Hamming* distance between the two species. We mention three special cases/variations of this problem.

**Hypercube Steiner trees.** The simplest case of the phylogeny problem is the case where each character can take only two values, say, 0 and 1. This case corresponds to the rectilinear Steiner tree problem where the coordinate in each dimension is either 0 or 1. The problem is also called the Steiner tree in hypercube problem, since it is identical to the Steiner tree problem on a hypercube graph. Foulds and Graham [157] showed that this problem is NP-hard, and it remains NP-hard when no more than two coordinates differ from 0 in each terminal [283]. Upper and lower bounds on the length of Steiner trees for the hypercube problem were given by Miller and Perkel [283], Miller and Pritkin [284] and Jiang, Miller and Pritkin [226].

**Wagner trees.** In a Wagner tree [382], each of the characters $\alpha_1\alpha_2\ldots\alpha_d$ that represent a species has a numerical value, and the difference between two species is the sum of differences of character values across all sites. A minimum-cost Wagner tree is therefore identical to a minimum rectilinear Steiner tree for a problem instance where each coordinate can take only a fixed set of values. Farris [152] formalised the problem and suggested algorithmic methods for solving the problem. A recent application of Wagner trees to understanding the development and progression of solid tumors, such as those associated with many forms of cancer, can be found in [101].

**Hamming distance trees.** In this problem each of the characters $\alpha_1\alpha_2\ldots\alpha_d$ is taken from an alphabet $A$, and the distance between two species is the Hamming distance (as described above). An exact algorithm was suggested by Althaus and Naujoks [14], and Althaus et al. [13] showed that the problem remains NP-hard even for dimension 3 (with an arbitrarily large alphabet).

## Exercises

**3.1.** Prove Lemma 3.1.

**3.2.** Prove Lemma 3.5.

**3.3.** Let $T$ be a full and fulsome minimum rectilinear Steiner tree spanning at least 3 terminals. Show (without using Lemma 3.6) that any slide of a straight edge of $T$ preserves the topology of $T$. Hence, show that the condition that $T$ contains at most one bent edge can be omitted from the statement of Lemma 3.6.

**3.4.** Prove Lemma 3.9.

**3.5.** Suppose we are given a full Steiner topology $\mathcal{T}$ with $n$ terminals. Assume that $\mathcal{T}$ has a caterpillar topology (see Section 3.1.2). Locate the Steiner points in $\mathcal{T}$ such that the resulting rectilinear tree has a Hwang form. Your algorithm should run in $O(n)$ time. [Hint: For each possible root in a Hwang form tree, attempt to construct a tree with either a horizontal or vertical long leg.]

**3.6.** Given a Hwang form full component $T$ spanning at most $4$ terminals, show that there always exists a minimum rectilinear spanning tree of length at most $3/2|T|_1$ that spans the same set of terminals.

**3.7.** Prove the claim given in the proof of Theorem 3.10: show that there always exists an $i \in \{1, \ldots, k-3\}$ such that $|s_i t_i| \leq |s_{i+2} t_{i+2}|$ and $|s_{i+1} t_{i+1}| \geq |s_{i+3} t_{i+3}|$. [Hint: Assume that the claim does not hold. For a type (i) tree show that the segments on each side of the long leg must have strictly increasing length as we move from the root to the corner point, and arrive at a contradiction. For a type (ii) use the same approach, but consider both the full component and its corner-flipped version.]

**3.8.** Prove Lemma 3.15. [Hint: Use Lemma 3.9 and the existence of corner-flipped Hwang forms.]

**3.9.** Show that $(1, 1, 1, 1)$ is the sole wire length vector for any set of 3 terminals, or equivalently, that the half-perimeter wire length is equal to the length of a minimum rectilinear Steiner tree for 3 terminals.

**3.10.** Prove Lemma 3.23.

**3.11.** Show that Algorithm 3.2 correctly constructs a minimum rectilinear Steiner tree $T$. In particular, show that in the **for** loop all possible forms for $T_i$ are correctly constructed.

**3.12.** Show that the Steiner depth for a full component that spans $n$ terminals is bounded by $O(\log n)$.

## Notes

[34]The more general definition of a cross, for any norm, is given in Chapter 1, Section 1.6.2. Clearly, the definition in this chapter is consistent with the more general definition.

[35]The Hwang form is named after Frank K. Hwang, who gave the characterisation in his seminal paper from 1976 [211]. A simpler proof of the Hwang form was given by Richards and Salowe [325] (see also Zachariasen [431]). In this book we give a proof that is based on the theory of canonical forms developed for general fixed orientation metrics in Chapter 2.

[36] One of the first probabilistic bounds on the number of full components was given by Salowe and Warme [335] in 1995. They showed that for a set of randomly and uniformly distributed terminals, the expected number of full components spanning $k$ terminals, where $k$ is a constant, is $O(n^2)$. Also, for $k = \Omega(n)$ they proved that the expected number of full components spanning $k$ terminals is $O(1)$. Zachariasen [429] showed that the expected number of full components spanning $k$ terminals, and satisfying the rectangle property and a simplified bottleneck Steiner distance property, is $O(n(\log\log n)^{k-2})$. Finally, Fößmeier and Kaufmann [156] showed that the expected number of full components satisfying the tree star property is $O(n \cdot 2^{\sqrt{n \log n}})$ with high probability.

[37]The first computer code to compute minimum rectilinear Steiner trees appears to have been developed by Yang and Wing [423] in 1971. The algorithm used a reduction to the Hanan grid, and solved the graph problem using a branch-and-bound algorithm for graphs. The largest solved problem had 9 terminals, and it was solved on a reduced Hanan grid with 20 vertices and 31 edges — pruned using a path-convex hull approach. In a follow-up paper from 1972, Yang and Wing [424] mentioned that by using the Dreyfus and Wagner dynamic programming algorithm [132], it should be possible to solve problems with 14 terminals in a couple of minutes, but problems with 18 terminals would require more than a day of computing time. In 1989, Sirodenko [351] presented an approach that allowed problems with up to 11 terminals to be solved. In the early 1990s, a number of papers appeared. Ganley and Cohoon [162] presented a dynamic programming algorithm that made it possible to solve problems with up to 16 terminals; an improved FST-based dynamic programming algorithm increased the range to 18 terminals [163]. Using a considerable engineering effort, Thomborson et al. [369] managed to solve problem instances with up to 23 terminals. Shortly after, Salowe and Warme [335] suggested a GeoSteiner approach, allowing the solution of problem instances with more than 30 terminals; Hetzel [197] pushed the range to 50 terminals. The real breakthrough occurred a few years later when Warme [388] in 1998 computed minimum rectilinear Steiner trees for problem instances with more than 1000 terminals.

[38]In the original papers on the FLUTE algorithm a minimal wire length vector is called a *potentially optimal wire length (POWL) vector*. We find that the prefix ‘minimal’ better characterises the properties of these vectors, namely that they are the ones that are locally minimal under domination.

[39]In addition to the references to the primary sources given in this section, more details on some of the results discussed here can also be found in Part III, Chapter 3 of Hwang et al. [213] and in the survey article of Thomas and Weng [367].

[40]A classical paper on the physical layout of circuits, including the design of printed circuit boards, is that of Soukup [359]. A more recent overview of mathematical methods for the physical layout of printed circuit boards can be found in [1].

[41]The literature on the physical design of integrated circuits is vast. Some fairly recent books and theses include — in chronological order — Lengauer [251], Kahng and Robins [231], Pecht and Wong [304], Sarrafzadeh and Wong [342], Gerez [174], Sait and Youssef [334], Sherwani [350], Vygen [380], Saxena et al. [343] and Kahng et al. [228]. Some early contributions on the routing problem are Yang and Wing [425] and Hightower [198]; more recent surveys can be found in Möhring et al. [287], Cong et al. [118], Peyer [306], Moffitt et al. [286], Moffitt [285], Müller [289], Robins and Zelikovsky [328], Alpert et al. [10] and Gester et al. [176]. The list of combinatorial problems in chip design recently compiled by Korte and Vygen [240] illustrates the challenges in the field. The authors consider the chip design problem to be one of the most important application areas in (discrete) mathematics. In particular, efficient algorithms are needed to handle problems with millions of modules and nets.

# Chapter 4

# Steiner Trees with Other Cost Functions and Constraints

In this chapter we look at Steiner tree problems that involve other cost functions and constraints (beyond those discussed in the first three chapters) but that still can be solved exactly by exploiting the geometric properties of minimal solutions. We focus particularly on four types of Steiner tree problems: the gradient-constrained Steiner tree problem, which serves as another example of an exactly solvable Steiner tree problem in a Minkowski plane with useful applications; the obstacle-avoiding Steiner tree problem, which is an important variation of the Steiner tree problem with applications in the physical design of microchips; bottleneck and other $k$-Steiner tree problems, where there is a given bound on the number of Steiner points; and Steiner tree problems optimising a cost associated with flow on the network.

## 4.1 The gradient-constrained Steiner tree problem

We begin by investigating an example of the Steiner tree problem in a Minkowski plane other than the Euclidean or fixed orientation planes. The most important normed Steiner tree problem outside of these cases is the gradient-constrained Steiner tree problem. This problem is interesting both from a mathematical point of view and because of its industrial applications in, for example, access design in underground mines. Based on these applications, we will begin with an intuitive description of the problem, and then show that it is an example of a Minkowski Steiner tree problem, as defined in Section 1.6.

There are, of course, many other norms that could be studied as cost functions for the Steiner tree problem (other than the ones considered in this section and the previous chapters of this book), but very few of them lead to nice geometric properties in the associated Steiner network that can be used to construct exact solutions.

### 4.1.1 Basic properties of gradient-constrained Steiner trees

Let $(p_x, p_y)$ denote the Cartesian coordinates of a point $p$ in the Euclidean plane. We assume that the $y$-axis is vertical. Then, for any two points $p$ and $q$ the *gradient* of $pq$ is defined here to be the slope of the line segment from $p$ to $q$, and is denoted by $g(pq)$. That is,

$$g(pq) = \frac{q_y - p_y}{q_x - p_x}.$$

Note that the gradient is independent of the orientation of $pq$, that is, $g(pq) = g(qp)$.

Given a differentiable curve, we define the gradient of the curve at any given point to be the gradient of the tangent to the curve at that point. This leads to the following definition.

**Definition [Gradient-constrained network]**: A geometric network $T = (V(T), E(T))$ embedded in the plane is said to be a *gradient-constrained network* with respect to a given positive constant $m$ if each edge $e \in E(T)$ is a piecewise differentiable curve such that the absolute value of the gradient at each differentiable point of $e$ is at most $m$, and $e$ is a shortest path between its endpoints under this constraint.

Suppose $pq$ is an edge of a gradient-constrained network embedded in the Euclidean plane, where the maximum permitted gradient $m$ is given. If $|g(pq)| \le m$, then $pq$ is a straight line joining $p$ and $q$, and is referred to as a *straight edge*. However, if $|g(pq)| > m$, then $pq$ cannot be represented as a straight line without violating the gradient constraint, but it can be represented by a zigzag line joining $p$ and $q$ with each segment having absolute gradient $m$. Such edges are referred to as *bent edges*.

The lengths of edges in a gradient-constrained tree can be measured in a special metric, called the *gradient metric*. Suppose $o$ is the origin and $p = (p_x, p_y)$ is a point in the plane. Define a *vertical metric* of the line $op$ to be $|op|_v = cp_y$ where $c$ is a given constant. Then the gradient metric can be defined in terms of the Euclidean metric and a vertical metric. The length of $op$ in the gradient metric is defined as follows:

$$|pq|_g = \begin{cases} |pq| = \sqrt{(q_x - p_x)^2 + (q_y - p_y)^2}, & \text{if } |g(pq)| \le m; \\ |pq|_v = \sqrt{1 + m^{-2}}|q_y - p_y|, & \text{if } |g(pq)| \ge m. \end{cases}$$

It is easily checked that this defines a metric, as it is the maximum of two metrics which are equal exactly when $|g(pq)| = m$. Note also that the gradient metric is convex, although it is not strictly convex. The unit ball for the gradient metric looks like a Euclidean unit ball with the top and bottom flattened (Figure 4.1).

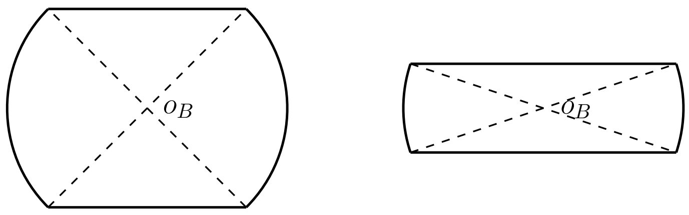


Figure 4.1: Unit balls $B$ for the gradient metric for $m = 1$ and $m = 1/3$, respectively. In each case the line segments from the origin $o_B$ to the non-differentiable points on the boundary of $B$ have gradients with absolute value $m$.

In formal terms, the associated Steiner tree problem can be stated as follows:

GRADIENT-CONSTRAINED STEINER TREE PROBLEM IN THE PLANE
**Given**: A set of points $N$ lying in the plane, and a gradient bound $m$ satisfying $0 < m \leq 1$.
**Find**: A geometric gradient-constrained network $T = (V(T), E(T))$ with respect to $m$, such that $N \subseteq V(T)$, and such that $|T|_g := \sum_{e \in E(T)} |e|_g$ is minimised.

As usual, the points in $N$ are referred to as *terminals*, and the nodes of $T$ not in $N$ (of degree $3$ or more) are referred to as *Steiner points*. A network $T$, solving this problem, is referred to as a *minimum $m$-constrained Steiner tree*. As in Chapters 2 and 3, it is convenient to think of the edges as being embedded as mimimum gradient-constrained paths in the Euclidean plane. Under such an embedding $|T|_g$ is equivalent to the Euclidean length of the embedded network. Like the rectilinear networks in Chapter 3, we can think of a gradient-constrained network embedded in the Euclidean plane as consisting of straight line *segments* that only intersect at their endpoints. There are two types of intersection points: the nodes of the network which consist of *terminals* (from the set $N$) and *Steiner points* (non-terminals with degree $3$ or more); and *corner points* (having degree $2$ where one incident segment has gradient $m$ and the other $-m$). Note that the angle between the two incident segments at a corner point is $2 \arctan m$.

A useful restriction on the problem (which is included in the definition above) is that the gradient bound $m$ satisfies $m \leq 1$. Let $\alpha = \arctan m$ be the angle of an edge of maximum gradient from the horizontal. The condition is equivalent to saying that $\alpha \leq \pi/4$. This condition (which is almost always satisfied in practical applications) is imposed largely for convenience. In particular, it immediately gives the following local property of $T$ at any node.

**Lemma 4.1** *Let $T$ be a minimum $m$-constrained Steiner tree (for $0 < m \leq 1$) in a*

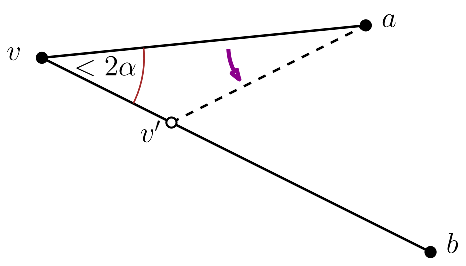


Figure 4.2: Replacing $av$ by $av'$, in the proof of Lemma 4.1, reduces the length of $T$ (illustrated here for $m = 1/2$).

*plane. Then the angle at which any two edges of $T$ meet at a vertex is at least $2\alpha$ (where $\alpha = \arctan m$).*

**Proof.** Suppose, on the contrary, that there exists a node $v$ of $T$ such that two of the edges incident with $v$, say $av$ and $bv$, meet at an angle less than $2\alpha$, and hence less than $\pi/2$. Clearly, at least one of the two edges, say $av$, has gradient of absolute value strictly less than $m$. Since $2\alpha < \pi/2$, the length of $T$ can now be strictly reduced by moving the point where this edge connects $a$ to $bv$ towards $b$ and away from $v$, to a new point $v'$ such that either $av'$ has absolute gradient $m$ or $v' = b$ (whichever occurs first), as illustrated in Figure 4.2. The shorter tree is still gradient-constrained, contradicting the minimality of $T$. ∎

It follows from Lemma 4.1 (and the gradient constraint) that all nodes of $T$ have degree at most $4$, and hence that all Steiner points of $T$ have degree $3$ or $4$. (This observation also follows from the more general results of Swanepoel [361], discussed in Section 1.6.2, on the maximum degree of Steiner points in a Minkowski plane.) Furthermore, the lemma has the following easy consequence, giving local geometric properties at each Steiner point.

**Corollary 4.2** *Let $s$ be a Steiner point of a minimum $m$-constrained Steiner tree $T$. Let $L_s$ be the vertical line through $s$. If $s$ is of degree $3$, then $s$ has two incident segments, one of gradient $m$ and the other of gradient $-m$, lying on one side of $L_s$, and a third incident segment lying on the other side of $L_s$. If $s$ is of degree $4$, then $s$ has two incident segments, one of gradient $m$ and the other of gradient $-m$, lying on each side of $L_s$.*

As in the previous chapters, we say that an $m$-constrained Steiner tree is *full* if every terminal has degree $1$. A minimum $m$-constrained Steiner tree $T$ can be uniquely decomposed into *full components*, meeting only at terminals, each of which is a full $m$-constrained Steiner tree on its corresponding set of terminals. The tree $T$ is said to be *fulsome* if it contains the maximum number of full components of any minimum $m$-constrained Steiner tree on the same set of terminals.

It is helpful to divide the study of the structure of full minimum $m$-constrained Steiner trees into two distinct cases: the first where all edges in the full tree have absolute gradient $\geq m$ and hence all segments of the tree have absolute gradient $m$; the second where there is at least one edge of gradient $k$ where $|k| < m$. Each of these two cases will be considered in turn.

**Case 1: All segments have absolute gradient $m$**

Let $T$ be a full minimum $m$-constrained Steiner tree such that every segment in $T$ has absolute gradient $m$. This is very similar to the situation for rectilinear Steiner trees. Indeed, since $T$ uses line segments with only two orientations, it follows from Theorem 3.21 and the proof of that theorem that we can assume $T$ has exactly the same canonical form as a minimum rectilinear Steiner tree. If we generalise the Hwang form (defined in Section 3.1.2 of Chapter 3) to any given pair of orientations, then we have the following result:

**Theorem 4.3** *Let $T$ be a full and fulsome minimum $m$-constrained Steiner tree that contains only segments of absolute gradient $m$. Then $T$ can be assumed to have Hwang form (with the two orientations corresponding to $m$ and $-m$).*

The consequence of this theorem is that for any terminal set $N$ there exists a minimum $m$-constrained Steiner tree on $N$ such that every full component either has a Hwang form or contains an edge with absolute gradient strictly less than $m$. The generation of all candidate full Steiner trees with all segments having absolute gradient $m$ (say, as part of a GeoSteiner approach) can be done efficiently in practice using the techniques discussed in Section 3.3. This part of the generation process is also made easier by the fact that only Hwang forms in which the corner point belongs to a bent edge need to be generated — otherwise we have have a full component belonging to Case 2, below.

**Case 2: There exists an edge with absolute gradient less than $m$**

For this case, let $T$ be a full minimum $m$-constrained Steiner tree such that $T$ contains a segment with gradient $k$ where $|k| < m$. Such a segment is said to have *intermediate* gradient. Clearly, the segment must be a straight edge since segments at a corner point have absolute gradient $m$. Furthermore, if a node $v$ is an endpoint of an edge in $T$ of intermediate gradient, then all edges of $T$ incident with $v$ are straight. This is an easy consequence of Lemma 4.1, since if there is a bent edge at $v$, then there exists an embedding of that edge such that two of the segments incident with $v$ have an angle between them of less than $2\alpha$, leading to a contradiction of the lemma; see Figure 4.3. By a similar argument it is also clear that $v$ has degree at most $3$.

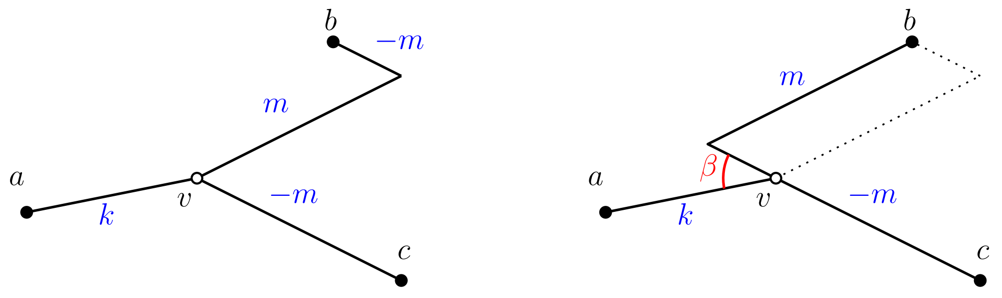


Figure 4.3: Suppose $av$ has intermediate gradient $k$ and the edge $(b, v)$ is a bent edge, as in the diagram on the left. Then the diagram on the right shows that there is an embedding of $(b, v)$ resulting in a meeting angle $\beta < 2\alpha$ at $v$, contradicting the minimality of $T$.

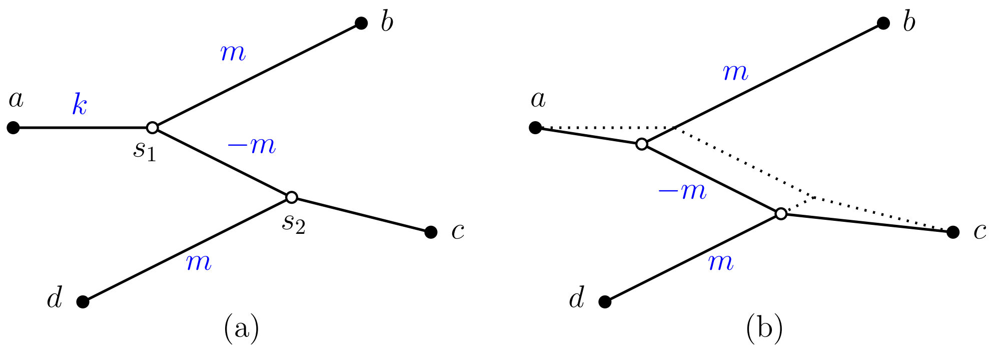


Figure 4.4: The effect of applying the potentially length-reducing perturbation $\zeta$. Diagram (a) shows the initial state, with the gradients of most edges indicated in blue. Diagram (b) shows the new network after applying the perturbation $\zeta$.

A key property of $T$ is that every edge in $T$ with intermediate gradient has the same gradient. This result mirrors Theorem 1.33 (for smooth metrics) and Theorem 2.11 (for polygonal metrics). More specifically, the following structure theorem holds.

**Theorem 4.4** *Let $T$ be a full and fulsome minimum $m$-constrained Steiner tree containing an edge of gradient $k$ where $|k| < m$. Then every Steiner point of $T$ is adjacent to three straight edges, one of gradient $m$, one of gradient $-m$ and one of gradient $k$.*

The idea of the proof of this theorem, which appears in [63], is to show that if there are two adjacent Steiner points in $T$ that do not satisfy the theorem, then there exists a simultaneous perturbation of both Steiner points that strictly reduces the length of $T$. This is illustrated in Figure 4.4. Suppose $T$ is the $m$-constrained tree with nodes $a, b, c, d$ shown in the figure. Here all edges have gradient $m$ or $-m$, apart from $as_1$ which has intermediate gradient $k$, and (possibly) $s_2c$ which may have any gradient (with absolute value $\leq m$) other than $k$. Each of the two Steiner points satisfies the conditions of Corollary 4.2, and it is straightforward to confirm that each Steiner point is locally minimal. In other words, no small movement or perturbation of $s_1$ or $s_2$ (which fixes the other Steiner point) can reduce the length of $T$. Consider, however, the 2-point perturbation $\zeta$ shown in the figure, where $s_1$ is perturbed along the line through $s_1b$ away from $b$, $s_2$ is perturbed along the line through $s_2d$ towards $d$, and $s_1s_2$ remains an edge of gradient $m$. Under this perturbation, the sum $|s_1b|+|s_1s_2|+|s_2d|$ clearly remains unchanged, but it is straightforward to show that either $\zeta$ or $-\zeta$ strictly reduces $|s_1a| + |s_2c|$, and hence the total length of the tree; see Exercise 4.1.

### 4.1.2 Construction of gradient-constrained Steiner trees

Using the properties in the previous section, we now show that there is a finite procedure for constructing a minimum $m$-constrained Steiner tree. Since there are only a finite number of possible Steiner topologies for any given cardinality of the terminal set, the problem can be reduced to that of constructing a minimum Steiner tree $T$ for a fixed full topology $\mathcal{T}$. If $T$ contains no edge with intermediate gradient (Case 1, in the previous section) then, like full minimum rectilinear Steiner trees, $T$ is a caterpillar (that is, the subgraph of $\mathcal{T}$ induced by the Steiner points is a single path) and $T$ can be constructed in linear time from $N$ and $\mathcal{T}$ — see Chapter 3 for more details.

If $T$ does contain an edge with intermediate gradient, then the construction is less obvious. The construction here makes use of Theorem 4.4. Recall, from Chapter 2, that for fixed orientation metrics, the usefulness of knowing that a full minimum Steiner tree uses only a single direction set lies in the fact that there are only a relatively small number of direction sets that need to be considered; in fact if there are $\sigma$ legal orientations (or equivalently $\sigma$ vertices on the polygonal unit ball), then the number of candidate direction sets that need to be considered is $\Theta(\sigma)$, and these can be identified in $\Theta(\sigma)$ time [74]. For the gradient metric, however, the 'direction set' at each Steiner point includes an unknown intermediate direction $k$ which belongs to the uncountable set $(-m, m)$. This potential difficulty in determining the intermediate direction has been resolved in [63], where it is shown that for any full minimum $m$-constrained Steiner tree there is a simple formula for computing the gradient of the intermediate edge in terms of the coordinates of the terminals and information about the topology of the tree. The correct statement of this result requires a number of definitions (presented below) based on an understanding of the structural properties of minimum trees.

A full Steiner topology $\mathcal{T}$ (with each Steiner point having degree 3) is referred to as a *labelled Steiner topology* if each edge of $\mathcal{T}$ is assigned a label $m$, $-m$ or $k$ such that the three edges meeting at each Steiner point all have different labels. A full minimum $m$-constrained Steiner tree $T$ is said to have a given labelled topology if there exists a

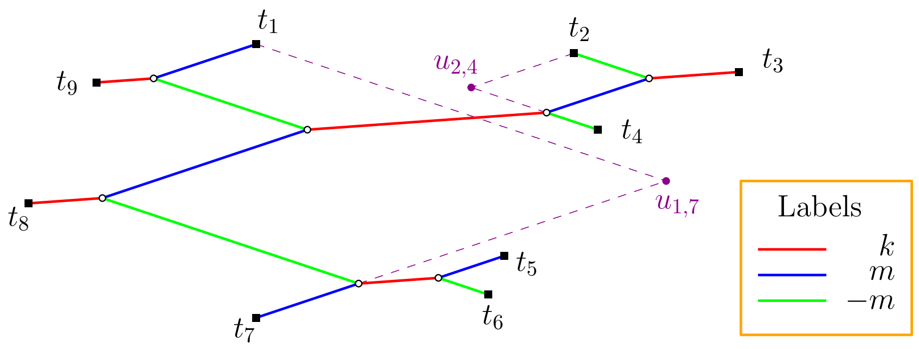


Figure 4.5: An example of a full minimum $m$-constrained Steiner tree on nine terminals (for the case where $m = 1/3$). The colour of each edge indicates its label. Note that the tree consists of three zigzag paths (in blue and green) connected to each other and the terminals $t_3$, $t_8$ and $t_9$ by red edges. The locations of two of the elements of $U(T)$ are shown in purple; the third element of $U(T)$, $u_{5,6}$, coincides with the Steiner point adjacent to $t_5$ and $t_6$.

real number $k$ with $|k| < m$ such that the gradient of each edge corresponds to the label on that edge. There are no restrictions on the topology of $T$ beyond each Steiner point having degree $3$, as it can be shown — using the threshold technique developed by Hwang and Weng [214] — that minimum $m$-constrained Steiner trees with any labelled Steiner topology can occur. However, there are some simple structural properties that $T$ must exhibit that allow us to compute $k$ without pre-knowledge of the locations of the Steiner points. In particular, $T$ can be viewed as being composed of pairs of terminals with zigzag paths (known as *$m$-zigzags*) between them, where each edge has gradient $m$ or $-m$; these $m$-zigzags are connected to each other and other terminals by edges with gradient $k$. This is illustrated in Figure 4.5. Note that, given the labelled topology, we can identify which terminals are at the top and bottom of each zigzag by looking at the relative $y$-coordinates of the endpoint terminals.

It follows from Corollary 4.2 that the sequence of gradients in an $m$-zigzag is strictly alternating in sign, and that the path rises monotonically from the bottom terminal end-point to the top. Clearly, the set of $m$-zigzags in $T$ is well defined, and any two distinct $m$-zigzags are disjoint. We denote a given $m$-zigzag by $z(t_i, t_j)$ where $t_i$ is at the top and $t_j$ is at the bottom of the zigzag.

Given an $m$-zigzag $z(t_i, t_j)$, let $\varepsilon$ be the element of $\{-1, 1\}$ such that the edge of $T$ incident to $t_j$ has gradient $\varepsilon m$. Define the point $u_{i,j}$ to be the intersection of the line of gradient $\varepsilon m$ passing through $t_j$ and the line of gradient $-\varepsilon m$ passing through $t_i$. Let $U(T)$ be the set of all such points $u_{i,j}$ in $T$ (hence, the cardinality of $U(T)$ equals the number of $m$-zigzags in $T$). See Figure 4.5 for an example.

Let $L(T)$ be the set of terminals of $T$ such that for each $t_l \in L(T)$ either the edge of $T$ incident to $t_l$ has gradient $k$ or there exists an $m$-zigzag $z(t_l, t_a)$ such that the edge of $T$ incident to $t_l$ does not lie on the line segment $t_l u_{l,a}$. (In the example in Figure 4.5 $L(T)$ consists of the terminals $t_1, t_2, t_3, t_8$ and $t_9$, but does not include $t_5$.)

Note that the known orientation of each $m$-zigzag means that we can determine from the labelled topology of $T$ whether the edge incident to each $t_l \in L(T)$ lies to the left or right of $t_l$. So, for each $t_l \in L(T)$ we define

$$\varepsilon(t_l) = \begin{cases} \;\;1 & \text{if the edge of } T \text{ incident to } t_l \text{ lies to the left of } t_l, \\ -1 & \text{if the edge of } T \text{ incident to } t_l \text{ lies to the right of } t_l. \end{cases}$$

Similarly, for each $u_{i,j} \in U(T)$ we define

$$\varepsilon(u_{i,j}) = \begin{cases} \;\;1 & \text{if the edge of } T \text{ incident to } t_j \text{ lies to the left of } t_j, \\ -1 & \text{if the edge of } T \text{ incident to } t_j \text{ lies to the right of } t_j. \end{cases}$$

If we think of all points in the plane as being described by their Cartesian coordinates (with respect to a given origin), then using the above definitions we have the following theorem.

**Theorem 4.5** *[63] Let $T$ be a full minimum $m$-constrained Steiner tree containing an edge of gradient $k$, where $|k| < m$, and with a labelled Steiner topology. Let $U(T)$ and $L(T)$ be defined as above. Define the vector $(x, y)$ by the equation*

$$(x, y) = \sum_{t_l \in L(T)} \varepsilon(t_l) t_l + \sum_{u_{i,j} \in U(T)} \varepsilon(u_{i,j}) u_{i,j}.$$

*Then $k = y/x$.*

The proof is a fairly technical but straightforward inductive argument on the number of terminals of $T$ and is not given here. The base cases for the induction, where the number of terminals is either two or three, are however easy to verify (Exercise 4.2).

The above theorem means that for a labelled Steiner topology the corresponding $m$-constrained Steiner tree can be constructed in linear time. Since there are only a finite number of possible labellings for a given Steiner topology, it follows that Steiner trees for Case 2 can be constructed in finite time, as required.

Although an $m$-constrained Steiner tree can be constructed in linear time for a given labelled Steiner topology, it is not clear that the same thing is true for an unlabelled Steiner topology. If the label of one of the edges at a particular Steiner point in a Steiner topology is known, then there are two possibilities for the labelling of the other two edges. Hence, the number of possible labellings for a given full Steiner topology increases exponentially with the number of Steiner points (or terminals).

We know, however, from Chapters 1 and 2, that for a given terminal set and full Steiner topology a Euclidean Steiner tree can be constructed in linear time (Theorem 1.5) and a fixed orientation Steiner tree can also be constructed in linear time (Theorem 2.27), so it seems likely that a similar result is true for gradient-constrained Steiner trees.

**Conjecture 4.6** ***[Fixed topology gradient-constrained Steiner tree conjecture]*** *Suppose we are given a set of $n$ terminals $N$, a full Steiner topology $\mathcal{T}$ for that set of terminals and a gradient bound $m$. Then in polynomial time (in $n$) it is possible to either construct a full and fulsome $m$-constrained Steiner tree for $N$ with topology $\mathcal{T}$, or determine that no such tree exists.*

We conclude this section by briefly noting that the gradient-constrained Steiner tree problem is NP-hard, which suggests that there will always be families of instances for which any exact solution algorithm is unable to scale efficiently. The result is similar to the computational complexity results for the Euclidean Steiner tree problem (Section 1.3.3) and the $\lambda$-geometry Steiner tree problem (Section 2.5.3). As in those cases, it is possible to prove a somewhat stronger result, namely that the decision problem form of the (discretised) gradient-constrained Steiner tree problem is NP-complete even when the terminals are restricted to lying on two parallel lines.

The formal decision version of the problem is as follows:

PARALLEL LINES GRADIENT-CONSTRAINED STEINER TREE DECISION PROBLEM
**Instance**: A finite set of points $N$ lying on two parallel lines in the Euclidean plane, a gradient bound $m$ satisfying $0 < m \leq 1$ and a positive integer $L$.
**Question**: Is there an $m$-constrained Steiner tree $T$ with terminal set $N$ such that the length of $T$ is at most $L$?

The basic strategy for proving that this problem is NP-complete is very similar to that used in the Euclidean and $\lambda$-geometry cases. The idea is to show that solving any specific instance of the discretised parallel lines gradient-constrained Steiner tree decision problem depends on being able to solve the subset sum problem, that is, the problem of deciding for any given subset $S = \{d_1, \ldots, d_n\}$ of integers and integer $s$ whether there exists a subset $J \subseteq S$ such that $\sum_{i \in J} d_i = s$. The subset sum problem is known to be NP-complete, from which the theorem follows. The key to the construction of the instance encoding the subset sum problem is as follows: beginning with two vertical lines, place regularly spaced terminals on one line and closely spaced triples of terminals on the other in such a way that in a minimum $m$-constrained Steiner tree each triple will have a Steiner point adjacent to two of the terminals, and there will be a bent edge connecting the third terminal to one of that pair. For each triple there is a choice as to which pair of terminals has an adjacent Steiner point. The locations of all terminals are carefully chosen so that in a minimum $m$-constrained Steiner tree all edges with intermediate gradient are horizontal, and finding a choice of connection to each triple that makes all intermediate gradient edges horizontal is equivalent to solving the subset sum problem. This construction can easily be done in polynomial time from any instance of the subset sum problem, which completes the proof.

This leads to the following theorem.

**Theorem 4.7** *[63] The parallel lines gradient-constrained Steiner tree problem is NP-hard, for any given $m$ satisfying $0 < m \leq 1$; and hence the general gradient-constrained Steiner tree problem is NP-hard.*

Although the proof of this theorem was first outlined in [63], a more rigorous and complete proof can be found in [66].

### 4.1.3 Applications

One of the main practical motivations for studying the gradient-constrained Steiner tree problem lies in its application to the design of underground mines. Mining is a significant industry worldwide. Reducing the cost of mining operations is an important issue for mining companies in an extremely competitive and price-sensitive marketplace. For many years there have been well-developed methods for modelling and optimising the operation of *open-cut mines*, based originally on integer programming models developed in 1965 by Lerchs and Grossmann [252]. These methods have revolutionised the design of open-pit mines, allowing mining engineers and planners to manipulate and visualise data associated with optimal designs, and perform reliable risk analysis.

Until recently, no such methods were available for the design and planning of underground mining operations. A number of research groups, however, are now in the process of designing and developing an optimisation tool for underground mines. An example is the access optimisation tool, *Decline Optimisation Tool* (DOT), which has been developed at the University of Melbourne. This application efficiently determines a near-optimal (where optimal means least-cost) underground mining network servicing a given set of points associated with an ore body [48].

Several constraints, including a gradient constraint, are typically imposed on the geometry of the tunnels in the network, to ensure that the tunnels are navigable by haulage trucks. The associated gradient bound $m$ is usually about $1/7$, since this is typical of the maximum gradient at which a fully laden haulage vehicle can safely operate.

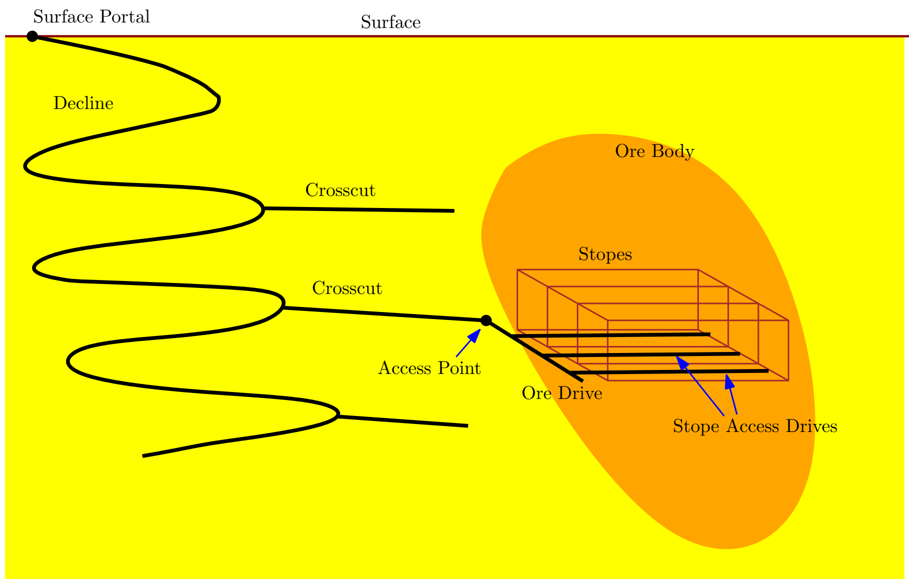


Figure 4.6: Schematic diagram showing some of the key elements of the access network for an underground mine, including a decline, crosscuts, and some stopes and part of the level layout (operational access to the stopes) on a single level.

## Modelling underground mining networks

An underground mine[42] consists of a series of interconnecting tunnels, ore passes (near-vertical chutes down which ore is dropped) and vertical shafts (used to hoist ore up to the surface). Its purpose is to allow extraction of ore containing valuable minerals (such as gold, silver, lead, zinc and copper) from underground locations to a predetermined surface portal (or breakout point), from where it is transported to a processing mill.

Ore zones (or stopes) are identified by geological tests such as surface and infill drilling. From this information, mining engineers can determine suitable draw points, which are the locations on the boundary of each stope from which the ore is accessed. The ore is then excavated using one of a number of possible mining methods (such as stoping, caving, room and pillar, etc.) and is transported to the surface via an access network using large haulage trucks. The access network is primarily composed of declines, which are the main traffic corridors for the haulage trucks, and crosscuts coming off the declines which give direct access to the stopes. See Figure 4.6.

Given that the laden trucks must be able to traverse the ramps, the following important constraints must be imposed on the design of the mine:

- Ramps must have absolute gradient not greater than some constant $m$. This constant is typically in the range 1:9 to 1:7 depending on equipment specifications.

- Ramps must satisfy a minimum turning radius (typically in the range of 15–30 metres), so that they are navigable by the trucks.
- Ramps must avoid certain no-go regions such as the interior of an ore body or other ramps in the mine.

A set of tunnels satisfying these constraints and interconnecting the draw points and a point at the surface can be modelled by a mathematical network $T$, where the draw points correspond to fixed vertices and the tunnels correspond to edges. The cost $C(T)$ associated with a network $T$ with a set $E$ of edges can be modelled by a function of the form

$$C(T) = \sum_{e \in E} (d + ht_e) l_e \tag{4.1}$$

where $d$ is a development cost rate (that is, the per-metre cost of tunnelling), $h$ is a haulage cost rate (per tonne·kilometre) associated with each edge of the network, $t_e$ is the total quantity of ore to be transported along an edge $e$ over the life of the mine, and $l_e$ is the length of $e$. The first term $\sum dl_e$ can be viewed as the total development cost of the mine, and the second component $\sum ht_e l_e$ as the total haulage cost associated with the mine over its life. If $h$ is set to zero, then the cost of the network is proportional to its length, and a network that optimises this objective function is a minimum length network of the type discussed in the previous two sections.

## Generalisations of the planar model

We now briefly mention some of the generalisations of the gradient-constrained Steiner tree problem in the plane that are most significant in the context of underground mine planning. In general it is important to recognise that the design of the access infrastructure in an underground mine is only one of many interconnected mine planning tasks that need to be undertaken when building and running an underground mine. These tasks range from optimal stope layout to high level strategic planning to access network design to scheduling of equipment and operations (amongst others). For a wider perspective on the range of opportunities for operations research in mine planning and the place of access network design, see [8] and [295].

The most natural generalisation of the gradient-constrained Steiner tree problem in the plane in the context of mine planning is to extend it to three dimensions. If one axis, say the $z$-axis, is assumed to be vertical, then the notions of gradient and gradient metric can easily be extended to the higher dimensional space. The associated Minkowski plane that correctly models length under the gradient metric becomes a Minkowski space, where the unit ball is a sphere flattened on the top and bottom (Figure 4.7).

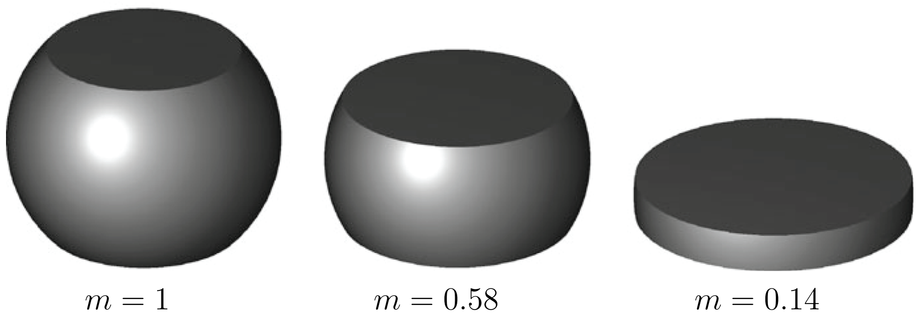


Figure 4.7: Three-dimensional unit balls for the $m$-constrained metric for a number of different values of the gradient bound $m$.

If the terminals in an instance of the three-dimensional gradient-constrained Steiner tree problem all lie in a vertical plane $\mathcal{P}$, then there exists a minimum Steiner tree $T$ that also lies in $\mathcal{P}$ and hence the problem reduces to the planar problem. This follows from the observation that the projection of $T$ onto $\mathcal{P}$ does not increase the length of any edge of $T$ (since the vertical displacement between the end-points remains the same and the horizontal displacement does not increase). This observation, however, is no longer true in general if the plane $\mathcal{P}$ is not restricted to being vertical.

There have been a number of papers studying the fundamental geometric properties of three-dimensional minimum gradient-constrained Steiner trees [60], [69], [61]. The construction of these Steiner trees is much more difficult than in the planar case. The problem can be approached by labelling the edges in a full topology to indicate whether each edge is bent, straight with absolute gradient $m$, or straight with absolute gradient less than $m$. It can then be shown that: Steiner points have degree $3$ or $4$; there are only a small number of different combinations of labellings that can occur on the edges incident to a Steiner point; and for each of these labellings the Steiner point can be determined in terms of the positions of the neighbouring vertices (though in some cases this has to be done numerically). These structural results have led to the development of a software tool called the *Underground Network Optimiser* (UNO) [49], [366].

The results described above focus on gradient-constrained networks which minimise length. While length-minimising networks optimise the total development (or infrastructure) cost associated with an underground mine, they do not account for the effects of haulage on the optimal design of the mine. Haulage costs can have a significant impact on the structure and layout of an underground mine, and considerable savings can be achieved by including both cost components as part of the optimisation objective. The general theory of networks minimising a cost function dependent not only on length but also on associated flows between terminals is discussed in more detail in Section 4.4. A detailed discussion of such minimum flow-dependent networks with a gradient constraint, in two or three dimensions, can be found in [71].

Two other important constraints, in the context of underground mine design, which significantly change the nature of the gradient-constrained Steiner problem are obstacle avoidance (which is the subject of the next section of this chapter) and the incorporation of a turning-circle constraint (see [50]).

## 4.2 Obstacle-avoiding Steiner trees

In all the previous sections of this book there has been an implicit assumption that the environment in which we are constructing an interconnection network is homogeneous; in other words, all regions of the plane are equally available to the network. In many applications, however, such an assumption may be an oversimplification; in particular, there may be known *obstacles*, or forbidden regions, which the network must avoid. In VLSI applications these obstacles may correspond to cells or modules containing transistors and other internal connections; in underground mining networks, as discussed in the previous section, the obstacles generally correspond to the ore bodies themselves and existing mine workings; in large-scale road networks the obstacles may be geographical obstructions, such as lakes or mountains.

The importance of obstacle avoidance in the construction of Steiner trees has been recognised in the literature, but most of the published work has focussed on heuristic methods and approximation algorithms.[43] Furthermore, those papers that have developed exact algorithms have only done so for the rectilinear or Euclidean metrics, and have tended to treat the underlying mathematical theory in a somewhat superficial way. In this section we aim to give a thorough and very general treatment of obstacle-avoiding Steiner trees, showing how the elegant underlying theory can be used to develop surprisingly efficient exact algorithms. The focus throughout this section will be on polygonal obstacles, though some discussion of smooth obstacles will also be given in Section 4.2.5.

### 4.2.1 Steiner trees with polygonal obstacles

We begin by investigating some of the fundamental properties of obstacle-avoiding Steiner trees in general Minkowski planes (which include the Euclidean and rectilinear planes).

#### Obstacle-avoiding paths in Minkowski planes

In order to understand the properties of obstacle-avoiding Steiner trees, we first investigate some of the basic properties of obstacle-avoiding paths.

Consider a Minkowski (or normed) plane containing a set $\Omega = \{\omega_1, \ldots, \omega_h\}$ of disjoint polygonal regions, which we refer to as *polygonal obstacles* (or simply *obstacles* if it is understood that the regions are polygonal). A vertex on the boundary of a polygonal obstacle is said to be a *convex vertex* if the interior angle at the vertex is less than $\pi$. For a given set of polygonal obstacles $\Omega$ we denote the set of convex vertices of all elements of $\Omega$ as $V_\Omega$.

**Definition [Obstacle-avoiding path]**: Given a set of obstacles in a Minkowski plane, a path between two points in this plane is said to be an *obstacle-avoiding path* if no point on the path lies in the interior of an obstacle.

The following fundamental property of obstacle-avoiding paths is well known in the Euclidean setting,[44] but is here proved more generally for Minkowski planes.

**Lemma 4.8** *For a given set of disjoint polygonal obstacles $\Omega$ in a given Minkowski plane, there exists a minimum length obstacle-avoiding path between two given points $p$ and $q$ (neither of which lies in the interior of an obstacle) which is a polygonal path whose inner vertices are elements of $V_\Omega$.*

**Proof.** Let $P$ be an obstacle-avoiding path between $p$ and $q$ with minimum length (under the given Minkowski norm). Suppose there exist distinct points $a$ and $b$ on $P$ such that $P_{ab}$, the section of $P$ between $a$ and $b$, is not a straight line segment, and the line segment $ab$ does not intersect the interior of any obstacle. If we replace such a subpath $P_{ab}$ by $ab$ in $P$, then we call this *shortcutting* $P$. Clearly shortcutting $P$ strictly reduces the Euclidean length of $P$ and does not increase its length under the given Minkowski norm (by the triangle inequality). We will show that if $P$ is not a polygonal path whose inner vertices are elements of $V_\Omega$, then we can shortcut $P$, and hence we can prove the lemma by choosing $P$ to have minimal Euclidean length.

Let $x$ be a point on $P$ that is not on the boundary of an obstacle in $\Omega$. Since $x$ is in the interior of obstacle-free space, there exists a disc of positive radius centred at $x$ that is completely contained in the free space, as illustrated in Figure 4.8(a). If the part of $P$ inside the disc is not a straight line segment then it can be replaced by a straight line segment connecting the two points where $P$ enters the disc and leaves the disc, shortcutting $P$. Hence, the path $P$ with minimal Euclidean length is polygonal with interior vertices on the obstacles in $\Omega$. By a similar argument, if any of these interior vertices of $P$ is not an element of $V_\Omega$, then again the Euclidean length of $P$ can be strictly reduced by shortcutting in the vicinity of that vertex (as illustrated in Figure 4.8(b)). ∎

The polygonal paths resulting from this lemma lead naturally to the concept of a visibility graph.

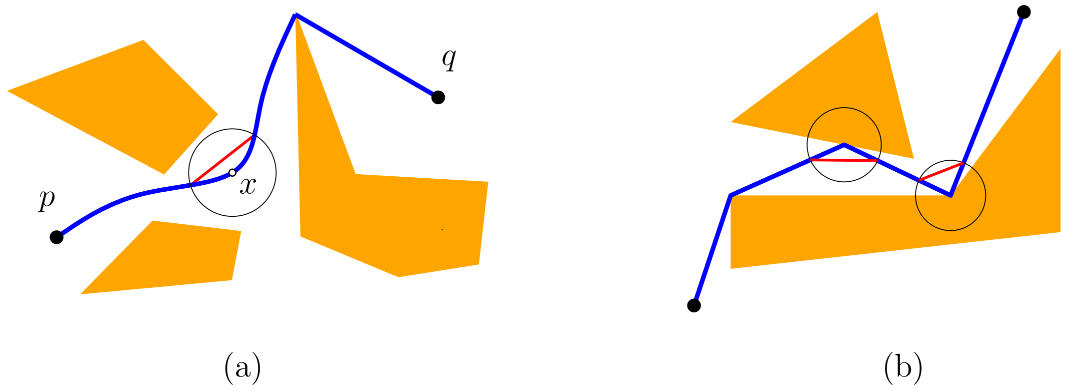


Figure 4.8: In (a), the path $P$ between $p$ and $q$ (shown in blue) can be shortcut in the vicinity of $x$ via the line segment shown in red. In (b), the polygonal path can be shortcut (as shown) at each of its interior vertices that meet an obstacle at a point not in $V_\Omega$.

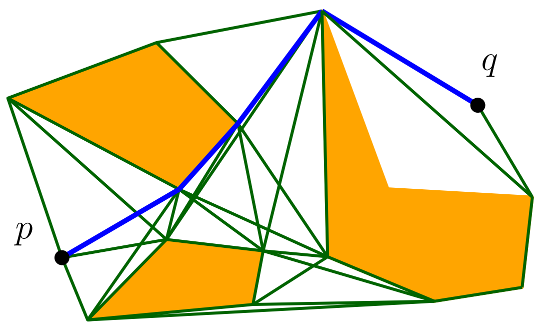


Figure 4.9: The visibility graph for $p$, $q$ and the convex vertices of the three obstacles from Figure 4.8(a). A minimum length obstacle-avoiding path between $p$ and $q$ (for the Euclidean metric) is indicated in blue.

**Definition [Visibility graph]**: Given a finite set of points $N$ and a finite set of polygonal obstacles $\Omega$ in the plane, we define the *visibility graph* for $N$ and $\Omega$ to be the network $G_{N,\Omega} = (V(G_{N,\Omega}), E(G_{N,\Omega}))$, where $V(G_{N,\Omega}) = N \cup V_\Omega$ and where $E(G_{N,\Omega})$ is the set of all line segments between elements of $V(G_{N,\Omega})$ that do not intersect the interior of any obstacle in $\Omega$.

It immediately follows from Lemma 4.8 that for a given set of obstacles $\Omega$ there exists a minimum length obstacle-avoiding path between two given points $p$ and $q$ that is a subpath of the visibility graph for $\{p, q\}$ and $\Omega$. An example of such a visibility graph is shown in Figure 4.9. Note that the visibility graph is independent of the Minkowski norm, but determining the minimum obstacle-avoiding subpath does depend on the norm (see Exercise 4.3), and can be computed, for example, using Dijkstra's algorithm [131].

An important property of visibility graphs is that they can be constructed efficiently.

Let $n$ be the cardinality of $V(G_{N,\Omega})$; it was independently shown by Welzl [393] and Asano et al. [17] that the visibility graph $G_{N,\Omega}$ can be constructed in time $O(n^2)$. Furthermore, for cases where the obstacles are densely packed, meaning that the visibility graph is relatively sparse, Ghosh and Mount [178] have developed an $O(m + n \log n)$-time algorithm for constructing $G_{N,\Omega}$, where $m$ is the cardinality of $E(G_{N,\Omega})$.

## Obstacle-avoiding Steiner trees in Minkowski planes

The concept of an obstacle-avoiding path can be easily generalised to that of an obstacle-avoiding network.

**Definition [Obstacle-avoiding network]**: Given a set of obstacles in a Minkowski plane, a network in this plane is said to be an *obstacle-avoiding network* if no point on the network (i.e., no vertex or interior point of an edge) lies in the interior of an obstacle.

As usual, our main interest is in optimal interconnection networks of this form. We now formally define the obstacle-avoiding Steiner tree problem. Recall, as in the previous chapters, that we denote by $\|\cdot\|$ the given Minkowski norm.

OBSTACLE-AVOIDING STEINER TREE PROBLEM IN THE PLANE
**Given**: A set of disjoint obstacles $\Omega$ and a set of points $N$ lying in a normed plane, such that no point in $N$ lies in the interior of an obstacle.
**Find**: A geometric obstacle-avoiding network $T = (V(T), E(T))$, such that $N \subseteq V(T)$, and such that $\|T\| := \sum_{e \in E(T)} \|e\|$ is minimised.

As in the obstacle-free case, $T$ necessarily has a tree topology, and hence we refer to $T$ as a *minimum obstacle-avoiding Steiner tree.* Note that any subpath of $T$ in which no internal vertex is a terminal and every internal vertex has degree $2$ is a minimum length obstacle-avoiding path between its endpoints. Hence, by Lemma 4.8, we can assume that every element of $V(T)$ of degree $2$ or less is an element of $N \cup V_\Omega$. More precisely, we have the following corollary.

**Corollary 4.9** *For any set of disjoint polygonal obstacles $\Omega$ and terminal set $N$ in a Minkowski plane there exists a minimum obstacle-avoiding Steiner tree $T$ with the following properties:*

1. *each vertex of $T$ is either: an element of $N$ (that is, a terminal), an element of $V_\Omega$, or a* Steiner point *of degree $3$ or more; and*

2. *the edges of $T$ are straight line segments, and each edge not incident to a Steiner point belongs to the visibility graph $G_{N,\Omega}$.*

For the remainder of this section we will assume that a minimum obstacle-avoiding Steiner tree in a Minkowski plane has the form described in Corollary 4.9 above. Associated with this characterisation of minimum obstacle-avoiding Steiner trees, we also have the following useful definitions.

**Definitions [Virtual terminals and full components]**: Given a set of obstacles $\Omega$ in a Minkowski plane, we refer to the set of points $V_\Omega$ as the associated set of *virtual terminals*. A minimum obstacle-avoiding Steiner tree $T$ in this Minkowski plane can be uniquely decomposed into *full components* (by splitting $T$ apart at each terminal and virtual terminal) where every terminal and every virtual terminal of each component has degree $1$ in that component.

With a slight abuse of definitions, we will refer to each full component of a minimum obstacle-avoiding Steiner tree $T$ as a *full minimum obstacle-avoiding Steiner tree* (although strictly speaking such a full component is not necessarily a minimum obstacle-avoiding Steiner tree, in fact it may have no terminals at all). As in the previous chapters, we also say that $T$ is *fulsome* if it contains the maximum number of full components amongst all minimum obstacle-avoiding Steiner trees on the same set of terminals.

We now observe that all the properties of Steiner points in Minkowski planes without obstacles (as discussed in Section 1.6), such as the degree and nature of the meeting angles, also apply to Steiner points in the obstacle-avoiding case. Essentially this follows from the fact that no meeting angle at a Steiner point is greater than $\pi$ (by the pointed configuration theorem, Theorem 1.26) and that the properties of Steiner points are all local properties using the local Steiner configuration. The idea is that in the obstacle-free case if a node of an interconnection network (not in $N$) is not a Steiner point, then there exists a perturbation of that node (which in the case of high degree nodes may also include a splitting or decomposition of the node into two or more nodes) that reduces the total length of the network. In the case with obstacles, there are two possibilities to consider: if such a node does not lie on the boundary of an obstacle, then we can apply exactly the same perturbation as in the obstacle-free case to reduce the length of the network; if the node does lie on the boundary of an obstacle $\omega \in \Omega$, then either it is a virtual terminal (that is, an element of $V_\Omega$) and hence not a candidate Steiner point, or else the length-reducing pertubation moves it away from the interior of $\omega$, and so it is not a Steiner point of the obstacle-avoiding network.

From the above discussion and the convexity of the length function, it follows that each full component of a minimum obstacle-avoiding Steiner tree $T$ is an obstacle-free full Steiner tree on the terminals and virtual terminals that it spans, in other words, is a relatively minimal tree for its Steiner topology. Hence, $T$ has the local properties of Steiner points presented in Section 1.6, giving the following result (Exercise 4.4).

**Theorem 4.10** *For any set of disjoint polygonal obstacles $\Omega$ and terminal set $N$ in a Minkowski plane there exists a minimum obstacle-avoiding Steiner tree $T$ such that each full component $T_i$ of $T$ is a full Steiner tree on the terminals and virtual terminals that it spans, and has the following properties:*

- *Every Steiner point in $T_i$ has degree at most $4$.*
- *If $T_i$ contains two or more Steiner points, then each Steiner point of $T_i$ has degree $3$.*
- *If the Minkowski plane is smooth, then each Steiner point of $T_i$ has degree $3$.*
- *If the Minkowski plane is smooth, then the edges of $T_i$ use at most three distinct orientations.*

Note that although each full component $T_i$, in Theorem 4.10, is a relatively minimal (obstacle-free) tree on the terminals and virtual terminals that it spans, it is not necessarily the globally minimum Steiner tree (over all Steiner topologies, in the obstacle-free plane). We will see an example of this in the discussion of Euclidean obstacle-avoiding Steiner trees in the next section.

### 4.2.2 Obstacle-avoiding Euclidean Steiner trees

We now look at some of the properties of minimum obstacle-avoiding Steiner trees in the Euclidean plane.[45] By Theorem 4.10, each full component $T_i$ of such a tree is a full Euclidean Steiner tree on the terminals and virtual terminals that it spans, and hence has the properties of a Euclidean Steiner tree (as discussed in Chapter 1). These properties include that the meeting angles at each Steiner point are $2\pi/3$ and the edges in $T_i$ use at most three different orientations differing by multiples of $2\pi/3$ (see Theorem 1.2).

Some examples of minimum obstacle-avoiding Steiner trees in the Euclidean plane are illustrated in Figure 4.10. The diagram on the left shows a minimum obstacle-avoiding Steiner tree interconnecting three terminals and avoiding a set of three obstacles. The obstacles have a total of 12 convex vertices which form the potential virtual terminals of the tree. The tree contains four full components, each of which is a Steiner tree on the spanned terminals and virtual terminals. The diagram on the right shows that a full minimum obstacle-avoiding Steiner tree is not necessarily a minimum Steiner tree. The minimum Steiner tree on the four terminals (shown in green) has length $16 + 15\sqrt{3} \approx 41.9808$, but is not obstacle-avoiding. The minimum obstacle-avoiding Steiner tree is

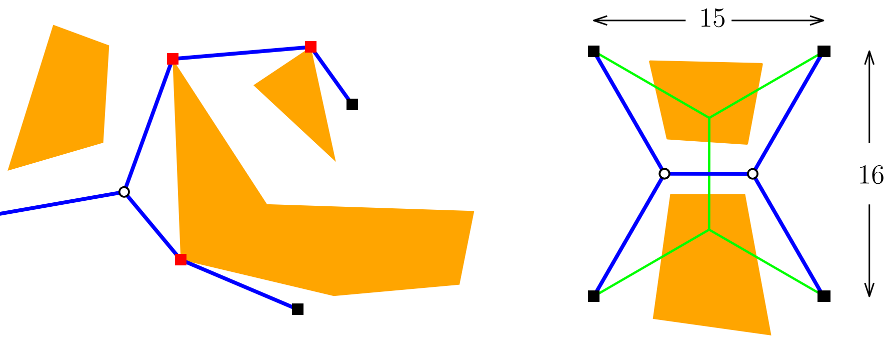


Figure 4.10: Two examples of minimum obstacle-avoiding Steiner trees in the Euclidean plane. The example on the left is composed of four full components; terminals are indicated by the black squares and virtual terminals by the red squares. The diagram on the right shows that full minimum obstacle-avoiding Steiner trees are not necessarily minimum Steiner trees, even though they must be Steiner trees. Here the four terminals lie on the corners of a $16 \times 15$ rectangle; the minimum Steiner tree is shown in green, while the slightly longer minimum obstacle-avoiding Steiner tree is shown in blue.

shown in blue; it is a Steiner tree, but with a different full topology, and has length $15 + 16\sqrt{3} \approx 42.7128$.

Theorem 4.10 is clearly sufficient to show that there is a finite algorithm for finding a minimum obstacle-avoiding Steiner tree for a given set of terminals $N$ and obstacles $\Omega$. For example, one could apply a GeoSteiner-type approach as follows:

1. Generate all full Steiner trees (with all possible full topologies) interconnecting subsets of points from the (finite) set $N \cup V_\Omega$;

2. Discard those full trees that intersect the interior of an obstacle in $\Omega$;

3. Compute all concatenations of the remaining full trees that span $N$ — such a concatenation with shortest total length is a minimum obstacle-avoiding Steiner tree.

A discussion of how to make this or other related approaches more efficient is given in Section 4.2.4 below.

The obstacle-avoiding Steiner tree problem in the Euclidean plane is NP-hard, since the problem without obstacles is already NP-hard (Theorem 1.18). It is clear in the above algorithm that the computational complexity depends both on the cardinality of $N$ and the cardinality of $V_\Omega$. Hence, it seems natural to ask the question of how efficiently a minimum Steiner tree can be constructed if we bound some, but not all, of the following elements: the number of terminals, the number of obstacles, or the number of convex vertices. We briefly survey a couple of results along these lines.

We first consider the case where there are three terminals and a single obstacle.

**Theorem 4.11** *[408] For any set of three terminals and a single convex polygonal obstacle with $k$ vertices in the Euclidean plane, a minimum obstacle-avoiding Steiner tree spanning the terminals can be constructed in $O(k)$ time.*

The result seems reasonable, since the set of convex vertices occurring in the interior of a shortest obstacle-avoiding path between a terminal and the Steiner point must constitute a set of adjacent vertices on the boundary of the obstacle. Winter and Smith [408] also show that if the obstacle is appropriately preprocessed in $O(k)$ time, then the above problem with the preprocessed convex polygonal obstacle can be solved in $O(\log k)$ time. This latter result is clearly useful for solving multiple three-terminal problems with the same convex obstacle. We also note that this result can be generalised to non-convex obstacles (by replacing the obstacle with its convex hull) as long as none of the terminals is located inside the convex hull.

More recently, in 2001, Weng and Smith [397] proved a more general result for a single obstacle and an unbounded number of terminals, but with a stronger set of assumptions. Given a minimum obstacle-avoiding Steiner tree $T$, define the *primitive topology* of $T$ to be the graph obtained from the topology of $T$ by iteratively replacing each pair of edges incident with a degree $2$ virtual terminal by a single edge (in other words, deleting all degree $2$ virtual terminals). Also, define a *convex path* of $T$ to be a subpath in $T$ from one terminal to another that either turns left at every interior node or turns right at every interior node. With these definitions we have the following result.

**Theorem 4.12** *[397] Given a set $N$ of $n$ terminals and a single polygonal obstacle with $k$ vertices in the Euclidean plane, suppose that the primitive topology for a minimum obstacle-avoiding Steiner tree $T$ spanning $N$ is known, and suppose further that all virtual terminals of $T$ are contained in a single convex path of $T$ between two terminals. Then $T$ can be constructed in time $O(n^2 + nk \log k)$, or in time $O(n^2 + n \log^2 k)$ if the obstacle is convex.*

### 4.2.3 Obstacle-avoiding fixed orientation and rectilinear Steiner trees

In this section, we look more specifically at properties of minimum obstacle-avoiding Steiner trees for fixed orientation metrics, including the rectilinear metric. The results in this section are largely new. Although there is some previous literature for the rectilinear version of this problem,[46] the results here are much more general, have simpler proofs, and subsume the earlier results.

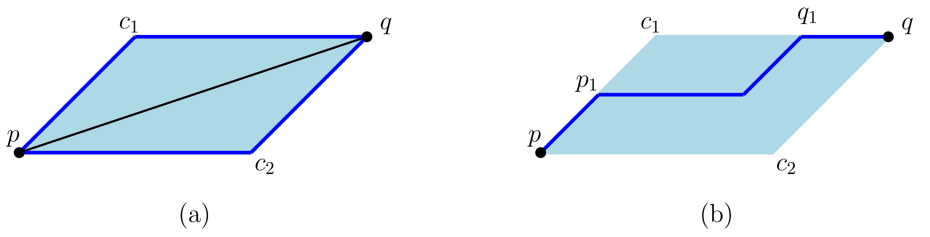


Figure 4.11: Diagram (a) shows two embeddings $pc_1q$ and $pc_2q$ using legal orientations for the edge $(p, q)$; these two geodesic paths form the boundary of the edge parallelogram of $(p, q)$. Diagram (b) illustrates a partial flip on $pc_1q$ resulting in a zigzag geodesic path embedding of $(p, q)$.

## Obstacle-avoiding Steiner trees for fixed orientation metrics

As discussed in Chapters 2 and 3, fixed orientation metrics correspond to cases where the unit ball is a centrally symmetric polygon (some examples of which are illustrated in Figure 2.4). Furthermore, these trees can be embedded in the *Euclidean* plane as trees composed of (possibly weighted) line segments restricted to legal orientations, which are the orientations of the vertices of the polygonal unit ball from the centre of the ball (see Section 2.1.2 for more details). As in the previous chapters, it is convenient to take this Euclidean point of view in order to better understand the geometry of the Steiner trees.

First we recall some properties of embeddings of a bent edge in the Euclidean plane. Let $(p, q)$ be a bent edge of a Steiner tree for a given fixed orientation metric. Then $(p, q)$ can be embedded as a pair of line segments in legal orientations in two different ways (Figure 4.11(a)); for each of the two embeddings there is a corresponding corner point ($c_1$ or $c_2$) where the two line segments meet. We describe the process of moving from one of these embeddings to the other as a *flip* (as in Chapter 3), and the parallelogram $pc_1qc_2$ will be referred to as the *edge parallelogram* of $(p, q)$. We can also perform a *partial flip* on the two line segments incident with a corner point; for example, if we choose any point $p_1$ on the interior of $pc_1$ and any point $q_1$ on the interior of $qc_1$, then performing a flip on the path $p_1c_1q_1$ (as in Figure 4.11(b)) gives a zigzag geodesic path embedding of $(p, q)$ with multiple corner points. By iteratively applying such partial flips we can create geodesic embedded paths with arbitrarily many corner points. It follows that the edge parallelogram of $(p, q)$ can be thought of as the union of all geodesic paths representing $(p, q)$ (compare Section 2.4.3).

To avoid some minor technical problems resulting from such embeddings in the presence of obstacles, we require the following definition.

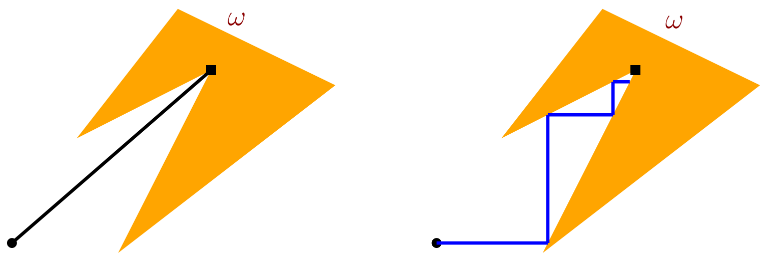


Figure 4.12: The diagram on the left shows an edge incident with a terminal at a restricted non-convex vertex of an obstacle $\omega$ with an obstacle-avoiding embedding in the Minkowski plane for the rectilinear metric. The diagram on the right indicates that there is no finite embedding of this edge in the Euclidean plane (using legal orientations).

**Definition [Corner-free terminal set]**: Given a fixed orientation metric and polygonal obstacle $\omega$ we say that a non-convex vertex $v$ on the boundary of $\omega$ is *restricted* if the orientations of the two boundary edges incident with $v$ (oriented outwards from $v$) lie on or between two adjacent legal orientations of the metric. A terminal set $N$ is said to be *corner-free* with respect to $\omega$ if no element of $N$ coincides with a restricted non-convex vertex of $\omega$. More generally, for a set of polygonal obstacles $\Omega$, $N$ is said to be *corner-free* with respect to $\Omega$ if it is corner-free with respect to each $\omega \in \Omega$.

Throughout this section we will require the terminal set to be corner-free with respect to the obstacle set. The reason for this is clear from Figure 4.12, which shows that although an edge of a tree $T$ incident with a terminal at a restricted non-convex vertex of $\omega$ can be embedded in the Minkowski plane, there may be no finite embedding of the same edge (using legal orientations) in the Euclidean plane.

The following lemma shows that once we impose this condition, obstacle-avoiding embeddings in the Euclidean plane can always be achieved for obstacle-avoiding trees in the corresponding Minkowski plane. In the proof of the lemma let "conv" denote the convex hull of a region or set of regions.

**Lemma 4.13** *For a Minkowski plane with a fixed orientation metric, suppose we are given: a set of disjoint polygonal obstacles $\Omega$, a set of terminals $N$ which is corner-free with respect to $\Omega$, and a fulsome minimum obstacle-avoiding Steiner tree $T$ for $N$. Then $T$ can be embedded in the Euclidean plane (with the same terminals and obstacles) so that it is obstacle-avoiding and no interior of a bent edge touches an obstacle.*

**Proof.** Let $(p, q)$ be any bent edge of $T$. We will show that there exists a minimum length embedding of $(p, q)$ in the Euclidean plane such that the interior of the embedded edge does not touch an obstacle, proving the lemma.

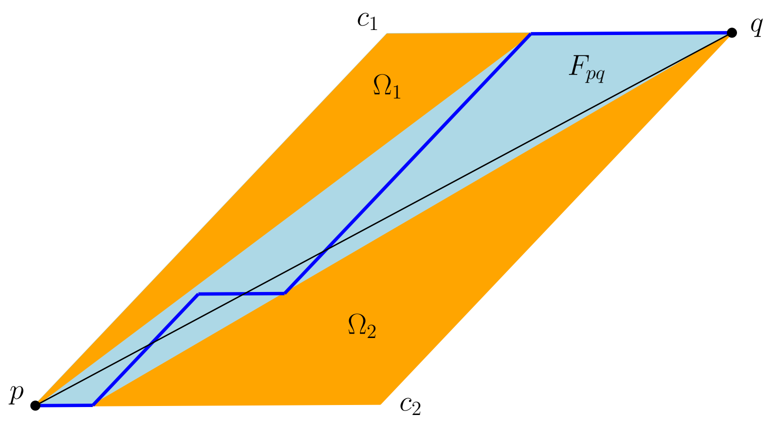


Figure 4.13: Construction of an obstacle-avoiding geodesic path between $p$ and $q$, shown in blue, using legal orientations.

Let $F_{pq} = pc_1qc_2$ be the edge parallelogram of $(p, q)$, where $c_1$ and $c_2$ are the two corner points; see Figure 4.13. The line segment $pq$ partitions $F_{pq}$ into two triangular sub-regions: $F_1 := \triangle pc_1q$ and $F_2 := \triangle qc_2p$.

Let $\Omega_F := \Omega \cap F_{pq}$. No component of $\Omega_F$ intersects the interior of both $F_1$ and $F_2$, since $(p, q)$ is obstacle-avoiding in the Minkowski plane. If either $\Omega \cap (F_1 \setminus \{p, q\}) = \emptyset$ or $\Omega \cap (F_2 \setminus \{p, q\}) = \emptyset$, then the lemma is true (since we can choose the embedding of $(p, q)$ to be $pc_1q$ or $pc_2q$, respectively). If, on the other hand, neither condition holds let $\Omega_1 :=$ conv$((\Omega_F \cup c_1) \cap (F_1 \setminus \{p, q\}))$ and $\Omega_2 :=$ conv$((\Omega_F \cup c_2) \cap (F_2 \setminus \{p, q\}))$. For each $i \in \{1, 2\}$, it is easy to see that there is no convex vertex of $\Omega$ on the boundary of $\Omega_i$ in the interior of $F_i$ (since otherwise we get a contradiction to fulsomeness); hence, $\Omega_i$ is a triangular region with one edge contained in $pc_i$ and another edge contained in $qc_i$.

Because the obstacles are disjoint and neither $p$ nor $q$ is a restricted non-convex vertex of an obstacle, it follows that at most one of $\Omega_1$ and $\Omega_2$ contains $p$, and similarly at most one of $\Omega_1$ and $\Omega_2$ contains $q$. Since $\Omega_1$ and $\Omega_2$ are disjoint it follows that there is a geodesic zigzag path from $p$ to $q$ not touching $\Omega_1$ and $\Omega_2$ (except at $p$ and $q$) which can be obtained from, say, $pc_1q$ by a finite series of flips and partial flips. This is illustrated in Figure 4.13. Finally note that the partial flips can be adjusted by an arbitrarily small amount so that none of the corner points touches $\Omega_1$ or $\Omega_2$. ∎

An easy corollary of the proof of Lemma 4.13 (see Exercise 4.5) is the following.

**Corollary 4.14** *For a given fixed orientation metric let $\Omega$ be a set of disjoint polygonal obstacles such that all boundary edges of the elements of $\Omega$ use legal orientations. Then the interior of the edge parallelogram for any bent edge of a fulsome minimum obstacle-avoiding Steiner tree is obstacle-free.*

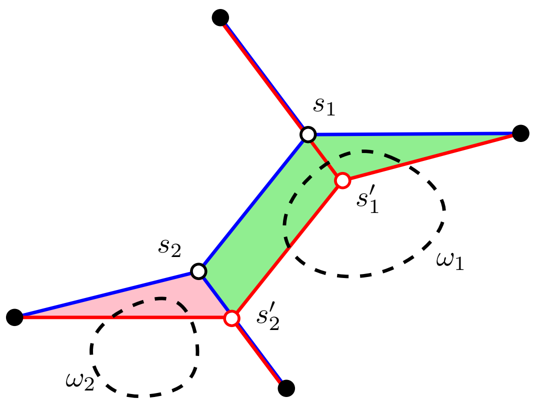


Figure 4.14: A 2-point fundamental zero-shift, where the original blue edges are obstacle-avoiding. In particular, the segment $s_2s_2'$ does not meet the interior of an obstacle.

In order to be able to efficiently construct minimum obstacle-avoiding Steiner trees, using, say, a GeoSteiner-type algorithm, we need to be able to show that the full components of the Steiner trees have the same canonical forms available to them as in the obstacle-free case. In particular, this means in the rectilinear case that each full component can be assumed to be in the Hwang form.

**Theorem 4.15** *For a given set of disjoint polygonal obstacles $\Omega$, let $T$ be a full and fulsome minimum obstacle-avoiding Steiner tree in a fixed orientation metric embedded in the Euclidean plane. Then for every canonical form of $T$ from the obstacle-free case (with the same topology) there exists an embedding of $T$ with that canonical form in the Euclidean plane that is obstacle-avoiding.*

**Proof.** We want to show that $T$ can have any canonical form that would be available to it (for its topology) if there were no obstacles. By Chapter 2, we know that we can move from a given full and fulsome Steiner tree to any canonical form via a sequence of fundamental zero-shifts; so it suffices to show that no fundamental zero-shift of $T$ can cause a part of $T$ to enter an obstacle. It is also sufficient to argue this in the Minkowski plane (with all edges in $T$ represented by straight line segments), since it follows from Lemma 4.13 that any such obstacle-avoiding Steiner tree has an obstacle-avoiding embedding in the Euclidean plane.

By Chapter 2, we know that there are two possible types of fundamental zero-shifts, 1-point and 2-point zero-shifts. We begin by considering any 2-point fundamental zero-shift, moving adjacent Steiner points $s_1$ and $s_2$ in $T$ to $s_1'$ and $s_2'$, respectively. Such a shift is shown in Figure 4.14, where the original (obstacle-avoiding) edges of $T$ are shown in blue, and the edges after the zero-shift are shown in red. Suppose, contrary to the theorem, that the red tree is not obstacle-avoiding. Then some connected section of the boundary of one of the obstacles in $\Omega$ enters the region between the blue edges and the red edge. More specifically, in terms of Figure 4.14, a connected section of the boundary of one of the obstacles, say $\omega_1$, enters either the region shaded in green or the region shaded in pink, but not both since the section of blue edge separating the two regions ($s_2 s_2'$ in the figure) is obstacle-avoiding. So suppose a connected section of the boundary of $\omega_1$ enters the green region. Since each meeting angle at $s_1'$ is at most $\pi$ it follows that any supporting line of the part of $\omega_1$ inside the green region touches a convex vertex of $\omega_1$ that lies within the green region. As $s_1$ moves to $s_1'$ under the zero-shift, one of the incident edges in $T$ must meet such a convex vertex as soon as it touches the boundary of $\omega_1$, giving a contradiction to fulsomeness. Similarly, if an obstacle enters the pink region (as illustrated by $\omega_2$ in the figure) we again get a contradiction to fulsomeness.

The same argument as above also applies for any 1-point zero-shift, giving the required result. ∎

## Reducing the set of virtual terminals for fixed orientation metrics

The efficiency of constructing minimum obstacle-avoiding Steiner trees can be further improved if we can reduce the number of virtual terminals that need to be considered for a given set of obstacles. The following definition suggests a method of reducing the complexity of obstacles.

**Definition [Transparent vertex]**: For a given polygonal obstacle $\omega$ let $v_i, v_{i+1}, v_{i+2}$ be three adjacent vertices on the boundary of $\omega$ such that $v_{i+1}$ is a convex vertex. Then $v_{i+1}$ is said to be a *transparent vertex* if:
(1) the line segment $v_i v_{i+2}$ lies entirely in $\omega$, and
(2) the path $v_i v_{i+1} v_{i+2}$ lies in the edge parallelogram of $v_i v_{i+2}$ (for a given fixed orientation metric).

Condition (2) of the above definition is equivalent to saying that the vectors $\overrightarrow{v_i v_{i+1}}$ and $\overrightarrow{v_{i+1} v_{i+2}}$ lie on or between two adjacent legal orientations. An example is shown in Figure 4.15(a) (illustrated for the rectilinear norm) where of the four convex vertices $v_1, v_2, v_3, v_4$ only $v_2$ is transparent; $v_1$ and $v_4$ fail condition (2), whereas $v_3$ satisfies condition (2) but fails condition (1).

The following lemma shows that if $v_{i+1}$ is transparent, then the path $v_i v_{i+1} v_{i+2}$ is geodesic, with respect to the fixed orientation metric (Exercise 4.6).

**Lemma 4.16** *For a given fixed orientation metric, with norm $\|\cdot\|$, let $v_{i+1}$ be a transparent vertex for a polygonal obstacle $\omega$, with neighbouring vertices $v_i$ and $v_{i+2}$. Then $\|v_i v_{i+1}\| + \|v_{i+1} v_{i+2}\| = \|v_i v_{i+2}\|$.*

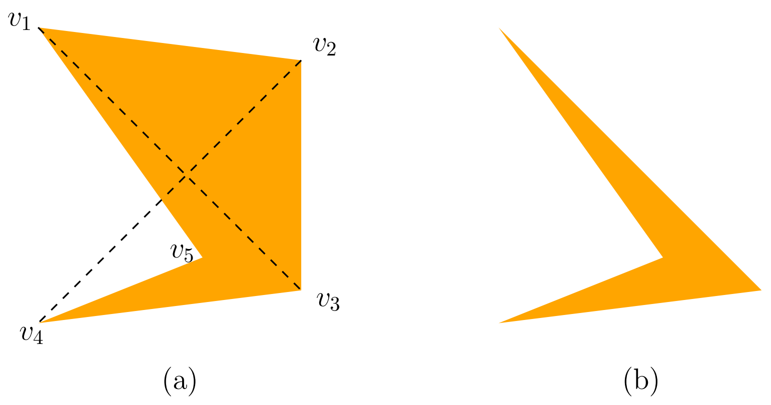


Figure 4.15: An example of a transparent vertex for an obstacle in the rectilinear plane. In diagram (a) the vertex $v_2$ is transparent. Diagram (b) shows the reduced obstacle obtained by removing vertex $v_2$.

If $v_{i+1}$ is a transparent vertex of an obstacle $\omega$, then we say that $\omega$ can be *reduced* by replacing the boundary edges $v_i v_{i+1}$ and $v_{i+1} v_{i+2}$ by the single edge $v_i v_{i+2}$ (or, in other words, deleting the triangular region $\triangle v_i v_{i+1} v_{i+2}$ from $\omega$). An example of a reduced obstacle is shown in Figure 4.15(b). In general, this reduction process can then be iterated by finding and removing a transparent vertex of the resulting obstacle. An obstacle is said to be *completely reduced* if after a series of reductions there are no more transparent vertices. Note that for a given obstacle $\omega$ a resulting completely reduced obstacle is not necessarily unique, but may depend on the sequence of reductions; see Figure 4.16.

**Theorem 4.17** *Given a fixed orientation metric and a set of disjoint polygonal obstacles $\Omega$, let $\Omega'$ be a corresponding set of completely reduced obstacles obtained from $\Omega$. Let $T$ be a fulsome minimum obstacle-avoiding Steiner tree with respect to $\Omega'$. Then there exists an embedding of $T$ in the Euclidean plane that is obstacle-avoiding with respect to $\Omega$.*

**Proof.** For a given polygonal obstacle $\omega$ let $\omega'$ be the reduced obstacle obtained by removing a single transparent vertex $v_{i+1}$ (where the neighbouring vertices on the boundary of $\omega$ are $v_i$ and $v_{i+2}$). Clearly, it suffices to show that if $\omega' \in \Omega'$ then there exists an embedding of $T$ in the Euclidean plane that is obstacle-avoiding with respect to $(\Omega' \setminus \{\omega'\}) \cup \{\omega\}$, since this result can then be iterated to obtain the statement of the theorem.

So suppose, in the Minkowski plane, that part of $T$ intersects the interior of the triangular region $\triangle v_i v_{i+1} v_{i+2}$. If this intersection consists of part of a single edge of $T$, meeting the path $v_i v_{i+1} v_{i+2}$ at, say, $p_1$ and $p_2$, then by Lemma 4.16 it follows that $p_1 p_2$ can be replaced by the path $p_1 v_{i+1} p_2$ without increasing the length of $T$, and that this modified form of $T$ can be embedded in the Euclidean plane so that it is obstacle-avoiding with respect to $(\Omega' \setminus \{\omega'\}) \cup \{\omega\}$ by Lemma 4.13.

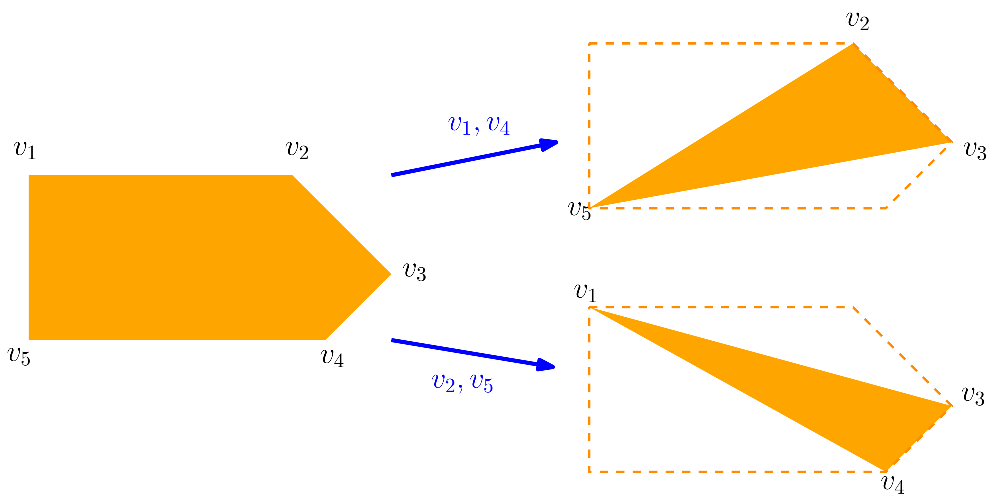


Figure 4.16: An example of an obstacle in the rectilinear plane for which there is not a unique completely reduced obstacle. The two different sequences of reductions, indicated in blue, result in different completely reduced obstacles.

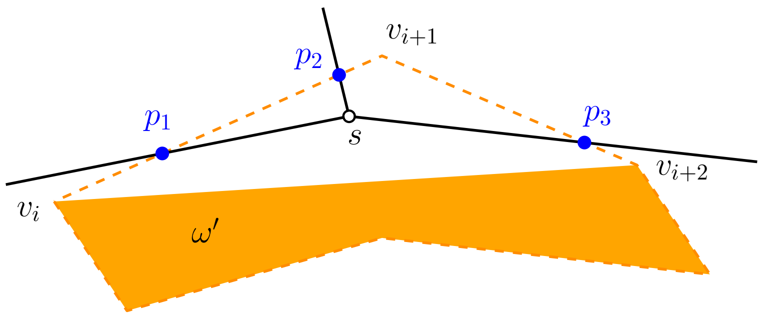


Figure 4.17: If there is a Steiner point $s$ of $T$ in the region $\triangle v_i v_{i+1} v_{i+2}$, where $v_{i+1}$ is transparent, then moving $s$ to $p_2$ strictly reduces the length of $T$.

Suppose on the other hand that the part of $T$ in the interior of $\triangle v_i v_{i+1} v_{i+2}$ contains a single Steiner point $s$, as illustrated in Figure 4.17. Then there are three (order labelled) points $p_1, p_2, p_3$ on the path $v_i v_{i+1} v_{i+2}$ where the three edges of $T$ incident with $s$ intersect that path. But, using Lemma 4.16 it is easy to see that $\|p_1 s\| + \|sp_3\| = \|p_1 p_2\| + \|p_2 p_3\|$. Hence, moving the Steiner point from $s$ to $p_2$ strictly reduces the length of $T$ (by at least $\|sp_2\|$) giving a contradiction to minimality. Finally, a similar argument applies if there are multiple Steiner points in the interior of $\triangle v_i v_{i+1} v_{i+2}$ . ∎

The consequence of Theorem 4.17 is that the complexity of computing a minimum obstacle-avoiding Steiner tree for a given fixed orientation metric can potentially be decreased by first completely reducing all obstacles.[47] This reduction process can be carried out in time $O(k)$ where $k$ is the total number of vertices in the set of obstacles (since transparency of a vertex can be checked in constant time, and reducing an obstacle cannot make a non-transparent vertex transparent).

It follows from the proof of Theorem 4.17 that we can obtain a minimum obstacle-avoiding Steiner tree for the unreduced problem instance simply by embedding each of the *bent edges* in the minimum obstacle-avoiding Steiner tree $T$ for the reduced instance appropriately.

### Reducing the set of virtual terminals for the rectilinear metric

We conclude this section by briefly looking at a couple of properties specific to the rectilinear plane. Suppose we are given a rectangular obstacle $\omega$ in the rectilinear plane (with vertical and horizontal edges). Note that initially all four vertices of $\omega$ are transparent, and a completely reduced obstacle for $\omega$ consists of a single line segment which is one of the diagonals of $\omega$ (where $T$ cannot cross the interior of such a line segment).

As an immediate corollary of Theorem 4.17 we have the following result.

**Theorem 4.18** *Let $\Omega$ be a set of disjoint rectangular obstacles with vertical and horizontal edges, and let $\Omega'$ be a corresponding set of line segments consisting of diagonals of the elements of $\Omega$. If $T$ is a fulsome rectilinear minimum obstacle-avoiding Steiner tree with respect to $\Omega'$, then there exists an embedding of $T$ in the Euclidean plane that is obstacle-avoiding with respect to $\Omega$.*

This result allows us to halve the number of virtual terminals that need to be considered. The restriction to rectangular obstacles is a natural one in many VLSI applications, and will be further discussed in Section 4.2.5.

For the second result, suppose that the set of obstacles consists of convex polygonal regions (with no restrictions on the orientations of edges). If some, or all, of the individual obstacles have large numbers of vertices, then the following theorem shows that the process of completely reducing the obstacles should substantially improve the complexity of computing a rectilinear minimum obstacle-avoiding Steiner tree.

**Theorem 4.19** *Let $\omega$ be a convex polygonal obstacle in the rectilinear plane. Then the corresponding completely reduced obstacle $\omega'$ contains at most four vertices.*

For the proof of this theorem see Exercise 4.7.

### 4.2.4 GeoSteiner algorithm

As in the obstacle-free case, the GeoSteiner approach is currently the most efficient exact algorithm for computing minimum obstacle-avoiding Steiner trees. The GeoSteiner approach has been adopted to the obstacle-avoiding Euclidean Steiner tree problem [435], and to the obstacle-avoiding rectilinear Steiner tree problem [206, 207, 208, 210]. In both cases, problem instances with hundreds of terminals have been solved to optimality.

In this section we describe the main modifications that are needed when applying the GeoSteiner approach to the obstacle-avoiding problem. We assume that the reader is familiar with the basic idea of the GeoSteiner algorithm; details are given in Section 1.4 (Chapter 1) on the Euclidean problem, in Section 2.6 (Chapter 2) on the fixed orientation problem, and in Section 3.3 (Chapter 3) on the rectilinear problem. In the GeoSteiner algorithm for the obstacle-avoiding problem, we first generate (obstacle-free) full Steiner trees (FSTs) spanning any subset of terminals and virtual terminals. Then in the FST concatenation phase, a subset of the generated FSTs is chosen such that the FSTs form a tree with minimum length that interconnects the terminals (and some subset of the virtual terminals).

Consider an (obstacle-free) full and fulsome Steiner tree $T$ that spans some subset of terminals and virtual terminals. From Theorem 4.15 we know that $T$ can be assumed to have any well-defined canonical form, e.g., the Hwang form in the rectilinear problem. This makes it possible to reuse core parts of the full Steiner tree generation algorithm from the obstacle-free case; this property contributes to making the approach very powerful even for the obstacle-avoiding case.

Recall the definition of a branch tree from Section 1.4.2 in Chapter 1. If we cut an edge $pq$ of a full Steiner tree $T$ at some point $c$, we obtain two branch trees: one rooted at $p$ having a stem (or ray) leaving $p$ along $pc$, and another rooted at $q$ having a stem (or ray) leaving $q$ along $qc$. In the FST generation algorithm, we combine branch trees to form larger branch trees. Branch trees of size 1 consist of a single (virtual) terminal having a stem leaving in one of the legal directions (for fixed orientation metrics) or any direction for other metrics.

The core pruning test in the obstacle-avoiding problem is that an edge $pq$ in a branch tree (or full Steiner tree) cannot intersect the interior of any obstacle in $\Omega$ — otherwise the branch tree (or full Steiner tree) is discarded. Most pruning tests for the obstacle-free problem can be used for the obstacle-avoiding problem in a slightly altered version. Below we briefly describe the modifications for an arbitrary Minkowski norm denoted by $\|\cdot\|$.

**Lune property.** Recall that a lune $\mathcal{L}(u, v)$ is defined as the set of points that are strictly within distance $\|uv\|$ of both $u$ and $v$. In the obstacle-avoiding problem, distances are shortest obstacle-avoiding paths under the given metric. If $uv$ is an edge in a minimum obstacle-avoiding Steiner tree, then $\mathcal{L}(u, v)$ cannot contain any terminal (Lemma 1.13 in Chapter 1).

**Bottleneck Steiner distance bound.** The bottleneck Steiner distance $\mathrm{BSD}(t_1, t_2)$ bounds the length of each edge on a path between terminals $t_1$ and $t_2$ in a minimum obstacle-avoiding Steiner tree (Lemma 1.14 in Chapter 1). The bottleneck Steiner distance $\mathrm{BSD}(t_1, t_2)$ is equal to the longest edge on the path between $t_1$ and $t_2$ in a minimum spanning tree for the terminals. In the obstacle-avoiding problem, distances between terminals are shortest obstacle-avoiding paths, and the minimum spanning tree is computed using these distances.

**Upper bounds.** Consider some branch tree $B$ with root $s$, and let $N(B)$ be the set of (virtual) terminals spanned by $B$. Since $B$ is assumed to be part of some minimum obstacle-avoiding Steiner tree, it must have minimum length. A number of different heuristics can be applied to provide an upper bound on the length of an obstacle-avoiding Steiner tree that interconnects $\{s\} \cup N(B)$. If any of these heuristics provide a tree that is shorter than $B$, then we can discard $B$.

The FST concatenation problem for the obstacle-avoiding problem can be solved either as a Steiner tree problem in a graph (Section 5.1 in Chapter 5) or as a Steiner tree problem in a hypergraph (Section 5.2.2 in Chapter 5).

### 4.2.5 Applications and extensions

Obstacle avoidance is one of the most important constraints in practical applications of the Steiner problem and related routing problems. In the design of microchips the obstacles correspond to the placement of cells or modules on the chip, or may represent pre-routed nets; see Section 3.6.2 in Chapter 3. In other infrastructure applications, such as the design of wireless sensor networks or roads or underground mines, the obstacles generally represent large physical obstructions or existing workings. These problems are closely related to constrained optimal path planning problems, which have applications in designing navigation systems for autonomous mobile robots and are becoming increasingly important in the design of computer games.

Throughout this section we have focussed on Steiner trees in the presence of *polygonal* obstacles. This constraint is a natural one in the context of microchip design, where the modules are not only polygonal, but also generally rectilinear. For more general applications, a restriction to polygonal obstacles can be justified by the ease with which general shapes can be accurately approximated by polygons [79]. The approach is also appropriate for problems based on real-world situations where the boundaries of the obstacles are determined by sampling, and then constructing a triangulated wireframe based on these sampled points (as is done in the determination of ore body and bad ground envelopes in underground mine design). If this is done in three dimensions, then for the corresponding two-dimensional problem the obstacle will be a cross section or projection of such a region, and hence will be polygonal.

Despite this, there are cases where it is worthwhile and interesting to consider smooth obstacles, and in some cases it is possible to obtain exact solutions. In particular, we now look briefly at the obstacle-avoiding Steiner tree problem in the Euclidean plane where the obstacles have circular boundaries. Let $\{C_i\}$ be a given set of circular obstacles and let $T$ be a minimum obstacle-avoiding Steiner tree for such a problem for a given set of terminals $N$. As before, we refer to a point at which a straight line segment of $T$ touches the boundary of an obstacle (tangentially) as a *virtual terminal*. Unlike the problem with polygonal obstacles, however, every point on the boundary of an obstacle $C_i$ is now a potential virtual terminal. Again we can think of $T$ as being composed of full components, where each full component is either an (obstacle-free) full Euclidean Steiner tree interconnecting terminals and virtual terminals, or is a boundary arc of one of the obstacles.

Weng [396] has shown that, given its Steiner topology, we can construct $T$ in linear time (in terms of the number of terminals and virtual terminals it spans) despite not knowing the locations of the virtual terminals beforehand. The key step to this construction is to show that any full component $T_i$ containing one or more Steiner points can be efficiently constructed using a generalisation of the Melzak-Hwang algorithm (see Section 1.2.1).

Recall that the main recursive step in the Melzak-Hwang algorithm involves replacing a pair of terminals adjacent to a single Steiner point by a so-called pseudo-terminal, which then acts as a terminal in later stages of the recursion. This can be generalised to the obstacle-avoiding case as follows. Suppose $T_i$ contains two virtual terminals $t_1, t_2$ adjacent to a single Steiner point $s$ and lying on the boundary of obstacles $C_1, C_2$ with centres $o_1, o_2$ and radii $r_1, r_2$, respectively. Let $v$ be the remaining vertex of $T_i$ adjacent to $s$. See Figure 4.18. Without loss of generality assume $r_2 \geq r_1$. Let $o_e$ be the third vertex of the equilateral triangle $\triangle o_1 o_2 o_e$, lying on the side of the line $\overline{t_1 t_2}$ opposite to $v$. Let $C_e$ be the circular region with centre $o_e$ and whose radius is $(r_2 - r_1)$ if $o_1$ and $o_2$ both lie on the same side of $\overline{t_1 t_2}$, or $(r_2 + r_1)$ if $o_1$ and $o_2$ lie on opposite sides of $\overline{t_1 t_2}$. Then Weng [396] shows that the line $\overline{vs}$ is tangent to $C_e$. Hence, we can replace the pair of obstacles $C_1$ and $C_2$ by the *pseudo-obstacle* $C_e$ to reduce the size of the problem. This is illustrated in Figure 4.18.

The important feature of this construction is that it can be performed without knowing the locations of the two virtual terminals on the boundaries of $C_1$ and $C_2$. It thus forms the basis for a generalised Melzak-Hwang algorithm, where the final step involves simply finding a line segment cotangent to the last two circular regions. Note that the same construction also applies if any of the virtual terminals in the description above is actually a proper terminal, by setting the corresponding radius to $0$.

A number of applications of this work on minimum obstacle-avoiding Steiner trees

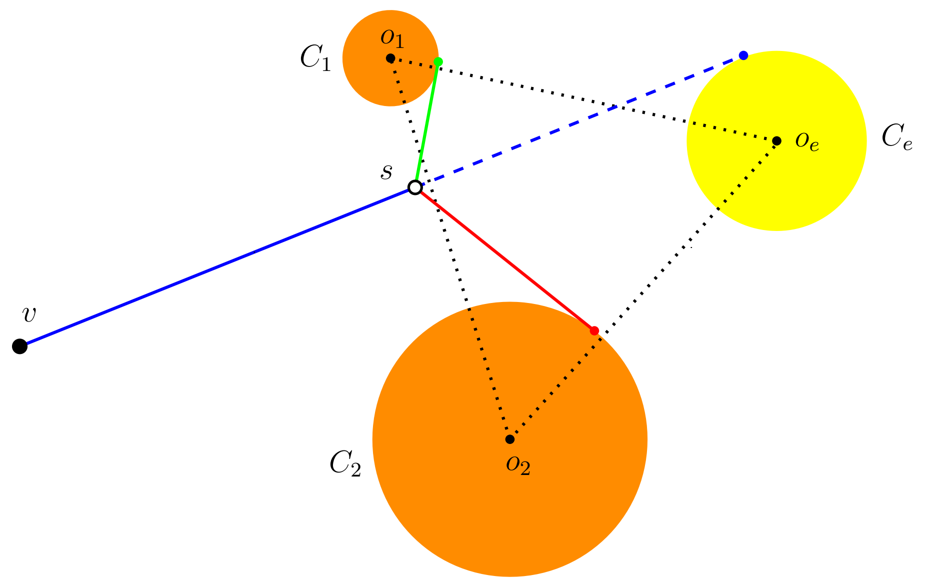


Figure 4.18: An illustration of how the basic recursive step in the Melzak-Hwang algorithm is generalised for obstacle-avoiding Steiner trees. Here the two circular obstacles $C_1$ with radius $r_1$ and $C_2$ with radius $r_2$ are replaced by a pseudo-obstacle $C_e$ with radius $r_2 - r_1$.

with circular obstacles have been suggested. Weng [395] shows that the method above can be adapted to optimally design networks that separate and surround circular regions. This has an industrial application in the chemical industry involving the design of safety infrastructure around storage tanks for hazardous liquid chemicals or petroleum products. To guard against possible spillage or other accidents the tanks need to be separated and surrounded by earthen dikes. The tanks and a minimum buffer region around each tank can be modelled as circular regions and the system of dikes can be modelled as a separating and surrounding network. The problem of minimising the construction and maintenance costs of the dikes then becomes a suitably constrained network optimisation problem.[48]

More generally, this approach can be used as an effective method for finding good heuristic solutions to the obstacle-avoiding Steiner tree problem with smoothly bounded obstacles, by approximating the boundary of each obstacle by a collection of circular arcs. An example of this approach is given in [396]. Weng [394] has also studied exact solutions for the Steiner tree problem with general smooth obstacles using a method based on differential calculus in the hexagonal coordinate system.

## 4.3 Bottleneck and general $k$-Steiner tree problems

One effective way of reducing the complexity of solving the Steiner tree problem is to place a bound on the number of Steiner points, independent of the number of terminals. Such a constraint results in problems that lie between the minimum Steiner tree problem and minimum spanning tree problem in computational complexity, and can generally be solved exactly in polynomial time (depending on the norm and the choice of cost function for the network). This restriction on the number of Steiner points is quite natural in some applications, where the addition of an extra node in the network may be expensive; in such situations the restriction on the number of Steiner points represents an absolute resource limitation.

A type of problem where an imposed bound on the number of Steiner points is not only natural but necessary is the geometric version of the bottleneck Steiner tree problem,[49] which is defined as follows:

BOTTLENECK $k$-STEINER TREE PROBLEM IN THE PLANE
**Given**: A set of points $N$ lying in a normed plane (with norm denoted by $\|\cdot\|$), and a positive integer $k$.
**Find**: A geometric network $T = (V(T), E(T))$, such that $N \subseteq V(T)$, $|V(T) \setminus N| \leq k$ and such that $\max\{\|e\| : e \in E(T)\}$ is minimised.

We refer to $T$ as a bottleneck $k$-Steiner tree, and we refer to $\max\{\|e\| : e \in E(T)\}$ as the *bottleneck length* of $T$. As in the previous chapters, we refer to the points in $V(T) \setminus N$ as *Steiner points*; however, for this problem Steiner points may have degree 2. Note that without a bound on the number of Steiner points the lengths of all edges in the network could be made arbitrarily small. We also note in the above definition that $T$, considered as a graph, can be assumed to be a tree: if $T$ contains a cycle, then any edge on the cycle can be removed without disconnecting the network or increasing the bottleneck length of $T$.

In this section we begin by developing an efficient exact algorithm for solving a very general class of $k$-Steiner tree problems. The algorithm makes use of a partitioning of the plane, known as the *overlaid oriented Dirichlet cell partition*. We then discuss stronger results that apply for the bottleneck Steiner tree problem in the rectilinear and Euclidean planes. We conclude the section by discussing applications of this work, both in wireless sensor networks and as a heuristic for solving the (unconstrained) Steiner tree problem.

### 4.3.1 The generalised $k$-Steiner tree problem

In this subsection, and the one that follows, we look at properties of $k$-Steiner trees, and use them to develop an efficient $O(n^{2k})$ algorithm for solving the associated Steiner tree problem over a wide range of norms and cost functions (where $n$ is the number of terminals and $k$ is the bound on the number of Steiner points). Our treatment here is based closely on the work of Brazil et al. [52], which generalises the approach developed by Georgakopoulos and Papadimitriou in [173].

#### Notation and fundamental definitions

We begin by establishing a formal definition for the generalised $k$-Steiner tree problem. The set of norms over which we define the problem is the following:

**Definition [Polygonal/elliptic norms]**: We define the *polygonal/elliptic norms* (*PE norms*) to be the set of norms $\| \cdot \|$ for which the unit ball $\{x : \|x\| \leq 1\}$ is either a polygonal or elliptic region.

The PE norms encompass the Euclidean norm (Chapter 1) and the general polygonal norms (Chapter 2), which include the rectilinear norm (Chapter 3).[50]

In order to define the set of possible objective functions for the generalised $k$-Steiner tree problem, we now set up our notation for this section. Given a set of terminals in the plane, $N = \{t_1, \ldots, t_n\}$, let $T$ be a geometric network interconnecting $N$ with tree topology $\mathcal{T}$. The tree $T$ contains $k'$ extra nodes (not in $N$) which we denote by $S = \{t_{n+1}, \ldots, t_{n+k'}\}$, and, as usual, refer to as the set of *Steiner points* of the network. Since $T$ is a tree, $E(T)$ contains $n + k' - 1$ edges which we denote $\{e_1, \ldots, e_{n+k'-1}\}$. Now let $\mathbf{e}_{\mathcal{T},N,S} = (\|e_1\|, \ldots, \|e_{n+k'-1}\|)$; i.e., the components of $\mathbf{e}_{\mathcal{T},N,S}$ are the edge lengths of such a network, for a given tree topology and a given set of embedded nodes. Such a vector is well defined up to the order of its components.

Let $\alpha : \mathbb{R}_+^{n+k'-1} \to \mathbb{R}$ be a symmetric function (i.e., independent of the order of the components of the vector on which it acts). We think of $\alpha$ as a cost function on a geometric tree network $T$, acting on the edges of $T$; we denote the cost of such a tree by $\|T\|_\alpha$. In other words, $\|T\|_\alpha = \alpha(\mathbf{e}_{\mathcal{T},N,S})$ is the cost of the network with topology $\mathcal{T}$ and nodes $N$ and $S$, and $\min\limits_{\mathcal{T},S} \alpha(\mathbf{e}_{\mathcal{T},N,S})$ is the minimum cost (with respect to $\alpha$) of any tree interconnecting the nodes $N$ and $k'$ other points, where $0 \leq k' \leq k$ for some given positive integer $k$.

For example, the standard network cost function we have considered so far, namely the sum of the lengths of the edges (under a given norm), is denoted by $\alpha_1$ and is defined by

$$\alpha_1(\mathbf{e}_{\mathcal{T},N,S}) := \sum_{i=1}^{n+k'-1} \|e_i\|.$$

For any positive real number $p$, we can generalise this to the power-$p$ cost function, $\alpha_p$, where the cost of each edge is the edge length raised to the power $p$ and the network cost is the sum of edges costs:

$$\alpha_p(\mathbf{e}_{\mathcal{T},N,S}) := \sum_{i=1}^{n+k'-1} \|e_i\|^p.$$

For the bottleneck cost function, $\alpha_\infty$, the cost of the network is the length of the longest edge:

$$\begin{aligned}\alpha_\infty(\mathbf{e}_{\mathcal{T},N,S}) &:= \max_{i=1,\ldots,n+k'-1} \|e_i\| \\ &= \lim_{p\to\infty} \left( \sum_{i=1}^{n+k'-1} \|e_i\|^p \right)^{1/p}.\end{aligned}$$

**Definition [$\ell_1$-optimisable function]**: We say that a symmetric function $\alpha = \alpha(\mathbf{e}_{\mathcal{T},N,S})$ is $\ell_1$-*optimisable* if and only if there exist $\mathcal{T}^*$ and $S^*$ such that $\alpha(\mathbf{e}_{\mathcal{T}^*,N,S^*}) = \min_{\mathcal{T},S} \alpha(\mathbf{e}_{\mathcal{T},N,S})$ and $\alpha_1(\mathbf{e}_{\mathcal{T}^*,N,S^*}) = \min_{\mathcal{T}} \alpha_1(\mathbf{e}_{\mathcal{T},N,S^*})$.

In other words, $\alpha$ is $\ell_1$-*optimisable* if for any given $N$ there exists a tree $T$ interconnecting $N$, with minimum cost with respect to $\alpha$, that is a minimum spanning tree on its complete set of nodes. It is easy to show that $\alpha_p$, for $p > 0$, and $\alpha_\infty$ are $\ell_1$-optimisable (Exercise 4.8).

GENERALISED $k$-STEINER TREE PROBLEM IN THE PLANE
**Given**: A set of points $N$ lying in the plane, a norm $\|\cdot\|$, a symmetric $\ell_1$-optimisable function $\alpha$, and a positive integer $k$.
**Find**: A geometric network $T = (V(T), E(T))$ with topology $\mathcal{T}$ and Steiner points $S$, such that $N \subseteq V(T)$, $|S| \leq k$, and such that $\|T\|_\alpha = \alpha(\mathbf{e}_{\mathcal{T},N,S})$ is minimised.

Note that $S = V(T) \setminus N$. We refer to $T$ as a *generalised minimum $k$-Steiner tree*. The lemma below shows that such a $T$ is equivalent in cost to *any* minimum spanning tree on $N \cup S$.

**Lemma 4.20** *Let $S$ be the set of Steiner points of a generalised minimum $k$-Steiner tree $T$ on $N$. Then every minimum spanning tree on $N \cup S$ is a generalised minimum $k$-Steiner tree on $N$.*

**Proof.** By the $\ell_1$-optimisability of $\alpha$ we may assume that $T$ is a minimum spanning tree on $N \cup S$. Let $T'$ be any other minimum spanning tree $N \cup S$. Using standard properties of minimum spanning trees (see, for example, the Swapping Algorithm in [249]) we can transform $T'$ to $T$ by a series of edge swaps, each of which replaces an edge with another of the same length. By the symmetry of $\alpha$ each such edge swap does not increase $\|T'\|_\alpha$. ∎

## The overlaid oriented Dirichlet cell partition

Throughout this subsection we assume that a PE norm $\|\cdot\|$ on $\mathbb{R}^2$ is given, with corresponding unit ball $B$; the boundary of $B$ is denoted by $\mathrm{bd}(B)$. Our aim is to describe the construction of the oriented Dirichlet cell (ODC) partition for any set $N$ of $n$ terminals embedded in this normed plane, and to show that it can be constructed in $O(n \log n)$ time. We also show that, with a time complexity of $O(n^2)$, multiple ODC partitions can be overlaid. This final partition is the so-called *overlaid ODC partition* (OODC partition), and is the key to devising an efficient algorithm.

The essential motivation behind an OODC partition is that it forms a partition of the plane with the following property: if $s$ is a Steiner point of $T$, then there is a set of up to six terminals, determined by the region of the partition in which $s$ lies, which are the only possible neighbouring terminals of $s$ in $T$. This partition can be constructed by overlaying a set of six carefully defined Voronoi diagrams. We will show that the total number of regions in this overlay is $O(n^2)$ and that the partition can be constructed in $O(n^2)$ time.

**Lemma 4.21** *There exist six points $\{y_i : i = 0, \ldots, 5\}$ on the boundary $\mathrm{bd}(B)$ of the unit ball $B$, with indices in anti-clockwise order around the boundary, such that for each pair of consecutive points $y_i, y_{i+1}$, we have $\|y_i y_{i+1}\| = 1$. Moreover, these six points are constructible.*

The lemma can be proved via a variation of the standard ruler and compass construction of the hexagon where $B$ plays the role of the circle in the construction. Given any point $y_0$ on $\mathrm{bd}(B)$ the idea is to construct a translation of $\mathrm{bd}(B)$ centred around $y_0$. The point $y_0$ along with the two intersection points of the two boundaries closest to $y_0$, constitute three of the six points. The remaining three points are constructed using the central symmetry of $B$ (Exercise 4.9). An example of such a set of six points is shown in Figure 4.19 for the unit ball of the rectilinear plane.

For any two directions $\phi_i$ and $\phi_j$ in the plane let $K(y, \phi_i, \phi_j)$ denote the cone formed from the union of all rays emanating from $y$ in direction $\phi$, for $\phi_i \leq \phi \leq \phi_j$. For each $y_i$ from Lemma 4.21 let $\theta_i$ be the direction of the ray $\overrightarrow{o_B y_i}$, where $o_B$ is the centre of $B$. We refer to the set $\{\theta_i\}$ of six directions as a *hexagonal direction set* of the norm $\|\cdot\|$. Note

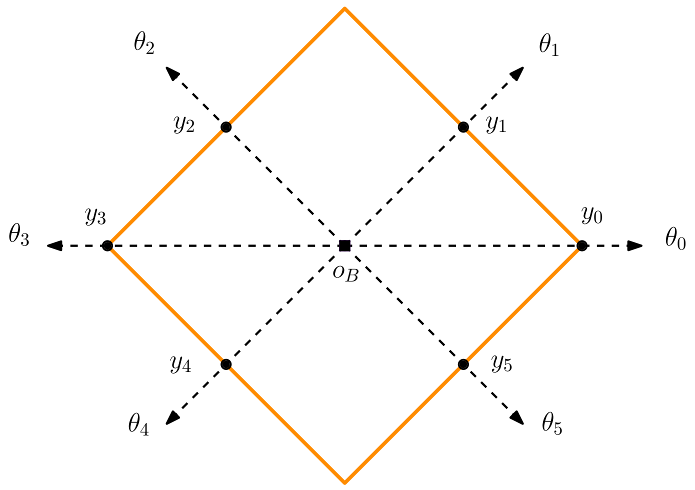


Figure 4.19: Six evenly spaced points $y_i$ on the boundary of the unit ball for the rectilinear plane, and the associated hexagonal direction set $\{\theta_i\}$.

that the $\{\theta_i\}$ are ordered in an anticlockwise manner, and two consecutive directions will be denoted by $\theta_i$ and $\theta_{i+1}$ (i.e. the $\mathrm{mod}\ 6$ notation will be omitted). Figure 4.19 shows a hexagonal direction set for the rectilinear plane.

**Lemma 4.22** *Let $x \in B$ with $x \neq o_B$ and $a, b \in \mathrm{bd}(B)$ such that the segments $ao_B$ and $bx$ intersect in a point. Then $\|ax\| \leq \|bx\|$.*

**Proof.** Let $p$ be the intersection of $ao_B$ and $bx$. Applying the triangle inequality to $\triangle pax$ and $\triangle pbo_B$, we obtain $\|ax\| + \|bo_B\| \leq \|bx\| + \|ao_B\|$ from which the lemma follows; see Figure 4.20. ▮

It is well known, and easy to prove, that in a Euclidean minimum spanning tree no angle between edges incident to a single point is less than $\pi/3$. This means that given a vertex $x$ of a minimum spanning tree $T$, if we partition the plane into six $\pi/3$ cones centred around $x$, each cone contains at most one neighbour of $x$ in $T$. The following lemma shows that a similar partitioning exists for any PE norm, based on the construction in Lemma 4.21.

**Lemma 4.23** *Given a plane with PE norm $\|\cdot\|$ and a corresponding hexagonal direction set $\{\theta_i\}$, let $y$ be any point in the plane. Then there exists a minimum spanning tree $T$ on $N \cup \{y\}$ with the following property: for each $i = 0, \ldots, 5$ there is at most one point of $N$ that is adjacent to $y$ in $T$ and lies in cone $K(y, \theta_i, \theta_{i+1})$, and this point is a closest terminal to $y$ in the cone.*

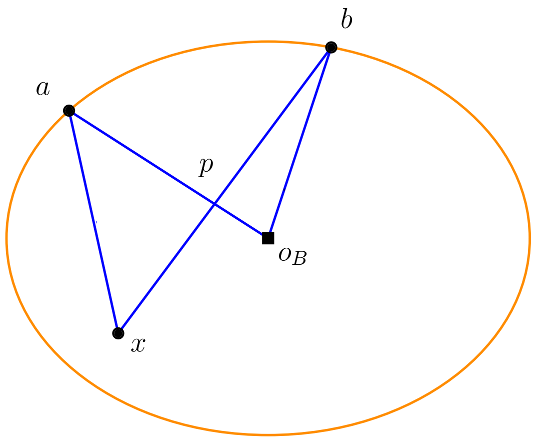


Figure 4.20: Illustration of the proof of Lemma 4.22 for the case where the unit ball is an elliptic region.

**Proof.** Let $T'$ be a minimum spanning tree on $N \cup \{y\}$. Let $t_0 \in N$ be a terminal in $K(y, \theta_i, \theta_{i+1})$ that is closest to $y$, and suppose that $(y, t_1) \in E(T')$ where $t_1 \in N$ is any other terminal in $K(y, \theta_i, \theta_{i+1})$. We show that we can replace the edge $(y, t_1)$ in $T'$ by either $(y, t_0)$ or $(t_1, t_0)$ so that the resulting tree is still a minimum spanning tree on $N \cup \{y\}$. From this, the statement of the lemma follows.

If the path in $T'$ connecting $y$ and $t_0$ passes through $t_1$, then clearly we can replace $(y, t_1)$ by $(y, t_0)$ without losing connectivity or increasing the length of $T'$. So, assume, on the other hand, that the path in $T'$ connecting $y$ and $t_0$ does not pass through $t_1$. We first prove the following claim.

**Claim.** $\|t_0t_1\| \leq \|yt_1\|$.

**Proof of Claim.** Without loss of generality, we assume that $y = o_B$, $\|o_Bt_1\| = 1$, and $K(o_B, \theta_i, \theta_{i+1})$ intersects the boundary of the unit ball $B$ in an arc from $a$ to $b$ (with $\|ab\| = 1$). We can also assume, without loss of generality, that $t_1$ lies on the same side of the line through $o_Bt_0$ as $b$. The convexity of $B$ implies that the line segments $o_Bt_1$ and $ab$ intersect; hence, by Lemma 4.22 we have

$$\|at_1\| \leq \|ab\| = 1. \tag{4.2}$$

We now prove the claim via two cases, illustrated in Figure 4.21.

Firstly, suppose $t_0$ and $o_B$ are on the same side of $at_1$ (including the case where $t_0$ lies on $at_1$). By inequality (4.2) $a$ lies in the unit ball centred at $t_1$, so, by convexity, $t_0$ also lies in this unit ball. Hence, $\|t_0t_1\| \leq 1$ as required. For the second case, suppose that $t_0$ and $o_B$ are on opposite sides of $at_1$. Let the ray from $t_1$ passing through $t_0$ intersect $B$ at $t_0'$. Then $t_0'$ and $o_B$ are also on opposite sides of $at_1$, and hence, by Lemma 4.22, $\|t_0t_1\| \leq \|t_0't_1\| \leq \|at_1\|$. Therefore, $\|t_0t_1\| \leq 1$ by inequality (4.2), completing the proof

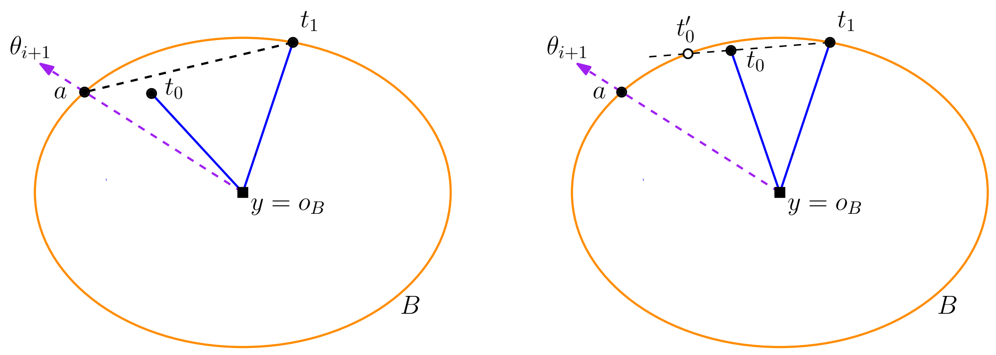


Figure 4.21: Constructions for each of the two cases in the proof of the claim in Lemma 4.23, illustrated for the case where the unit ball $B$ is an elliptic region. In each case $\|t_0t_1\| \leq \|o_Bt_1\|$.

of the claim.

By the claim we can now replace the edge $(y, t_1)$ by $(t_1, t_0)$ without losing connectivity or increasing the length of $T'$. ∎

**Definition [Oriented Dirichlet cell]**: Given a plane with PE norm $\|\cdot\|$ and a corresponding hexagonal direction set $\{\theta_i\}$, for each $i = 0, \ldots, 5$, we define the $i$th *oriented Dirichlet cell* (ODC) of $t \in N$ to be the set:

$$\{y \in \mathbb{R}^2 : \|ty\| = \min\{\|t'y\| : t' \in N \cap K(y, \theta_i, \theta_{i+1}).\}\}$$

In other words, the $i$th ODC of $t \in N$ is the set of all points $\{y\}$ whose closest terminal in the cone $K(y, \theta_i, \theta_{i+1})$ is $t$. We will show that the set of $i$th ODCs for all $N$, called the *$i$th ODC partition of $N$* is a type of Voronoi diagram.

For some background on Voronoi diagrams we refer the reader to the book by Okabe et al. [299] and the survey paper by Aurenhammer and Klein [18]. In the current context, a useful way of conceptualising Voronoi diagrams is the 'expanding waves' view of Chew and Drysdale [94]. If $n$ pebbles are dropped simultaneously into a pond, the places where wavefronts meet can be thought of as defining the Voronoi diagram on the $n$ points of impact. In the Euclidean case the wavefronts are circular, but in theory any closed convex curve $C$ bounding a region containing the origin $o$ can qualify as a wavefront and thereby define an abstract Voronoi diagram. We can think of this curve $C$ as representing the boundary of a unit ball. Here, $C$ defines a distance function $\delta_C$ on the plane where the distance of $o$ to a point $v$ is $|ov|/|ov_0|$ where $v_0$ is the point at which the ray $\overrightarrow{ov}$ intersects the boundary of $C$. (Note that $\delta_C$ is not necessarily a norm, since $C$ may not be centrally symmetric around $o$; however, being a distance function $\delta_C$ is invariant under translations.) For a given finite set of points $N$ the bisectors between elements

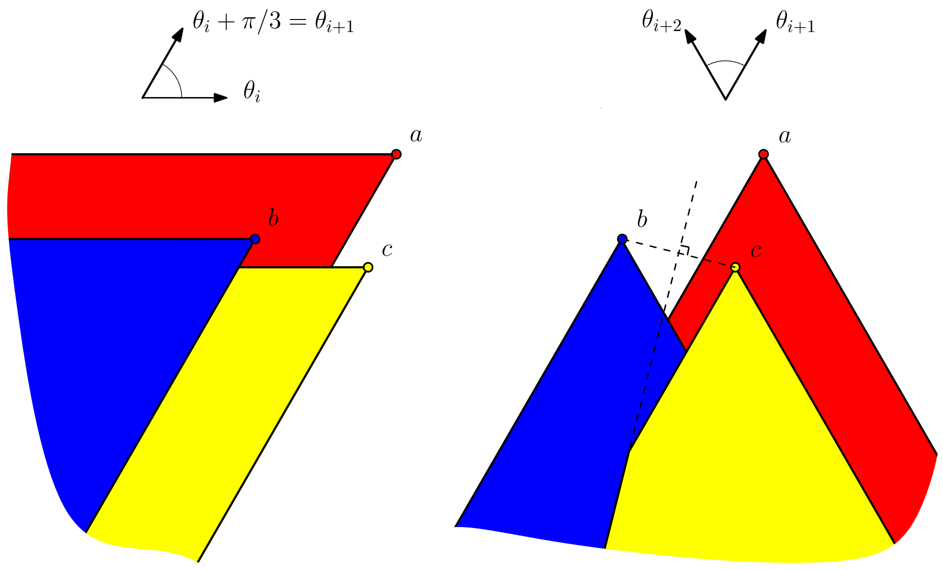


Figure 4.22: Two examples of ODC partitions (corresponding to different cones, shown at the top of the figure) for a set of three terminals in the Euclidean plane. Each region has the same colour as its corresponding terminal. Note, in the second example, that part of the boundary is determined by the bisector of $b$ and $c$.

of $N$ under the distance function $\delta_C$ (corresponding to the lines where wavefronts meet) determine the *Voronoi diagram of $N$ based on $C$*.

Here we slightly extend the definition of these abstract 'expanding waves' Voronoi diagrams from [94] by allowing for the possibility that $o$ may lie on $C$, rather than being restricted to the interior of $C$. If $o$ lies on $C$ this means that some points in $\mathbb{R}^2$ are infinitely distant from $o$. We can then define this generalised Voronoi diagram more formally as follows. Given a finite set of points $N$, for each $t \in N$ define a region $V_t = \{y \in \mathbb{R}^2 : \delta_C(t,y) < \infty$ and $\delta_C(t,y) = \min\{\delta_C(t',y) : t' \in N\}\}$. The set $\{V_t\}$ is the required Voronoi diagram based on $C$. Note that if $o$ lies on $C$, for the distance function, then the Voronoi diagram of $N$ based on $C$ does not cover the entire plane $\mathbb{R}^2$ since some points in the plane are infinitely distant from all elements of $N$.

Recall that $B$ is the unit ball of our given PE norm. It follows from the above discussion, that for each $i = 0, \ldots, 5$ the $i$th ODC partition of $N$ is equal to the Voronoi diagram of $N$ based on the boundary of the sector $B \cap K(o_B, \pi + \theta_i, \pi + \theta_{i+1})$. Some examples of such ODC partitions in the Euclidean plane are given in Figure 4.22.

We can now state the time and space complexity for calculating the $i$th ODC partition.

**Theorem 4.24** *Given a plane with PE norm $\|\cdot\|$ and a corresponding hexagonal direction set $\{\theta_i\}$, for any set of $n$ points $N$ the $i$th ODC partition of $N$ can be constructed in $O(n \log n)$ time and $O(n)$ space.*

**Proof.** [Sketch] The result follows immediately from a theorem of Chew and Drysdale [94] that states that the generalised Voronoi diagram of $n$ points based on a closed convex shape $C$ can be constructed in $O(n \log n)$ time and $O(n)$ space as long as the following operations can be performed in constant time:

1. Given two points, find the boundary where the two wavefronts meet.
2. Given two such boundaries, compute their intersection(s).

It is easy to see that these conditions hold for PE norms, since the bisector of two points for a PE norm consists of a bounded number of straight line segments. This is clear for the Euclidean and polygonal unit balls; for an elliptical unit ball it follows from the observation that an ellipse is a linear transformation of a circle, and hence a bisector for an elliptical norm is a linear transformation of a Euclidean bisector. ∎

By the *overlay* of two or more partitions we mean the union of the boundaries of all regions in the partitions, and the resulting regions from this union. This concept leads to the following definition.

**Definition [Overlaid oriented Dirichlet cell partition]**: Given a plane with PE norm $\| \cdot \|$ and a corresponding hexagonal direction set, the overlaid oriented Dirichlet cell partition (OODC partition) for a finite set of points $N$ is the overlay of the six $i$th ODC partitions for $N$.

An example of an overlay of two ODC partitions is given in Figure 4.23. (See also Exercise 4.10.)

**Theorem 4.25** *Given a plane with PE norm $\|\cdot\|$ and a corresponding hexagonal direction set $\{\theta_i\}$, for any set of $n$ points $N$ the OODC partition can be constructed in $O(n^2)$ time and contains $O(n^2)$ regions.*

**Proof.** [Sketch] This follows from a result of Georgakopoulos and Papadimitriou [173] showing that $q$ linear plane partitions, with $n$ regions in each partition, can be overlaid in $O(q^2n^2)$ time. The observation about the number of regions is also straightforward. ∎

Note that each region $R$ of an OODC partition is an intersection of at most six regions from different ODC partitions. Each of these ODC regions has a corresponding terminal $t_j$. Let $I_R$ be the set of up to six indices of the terminals corresponding to the

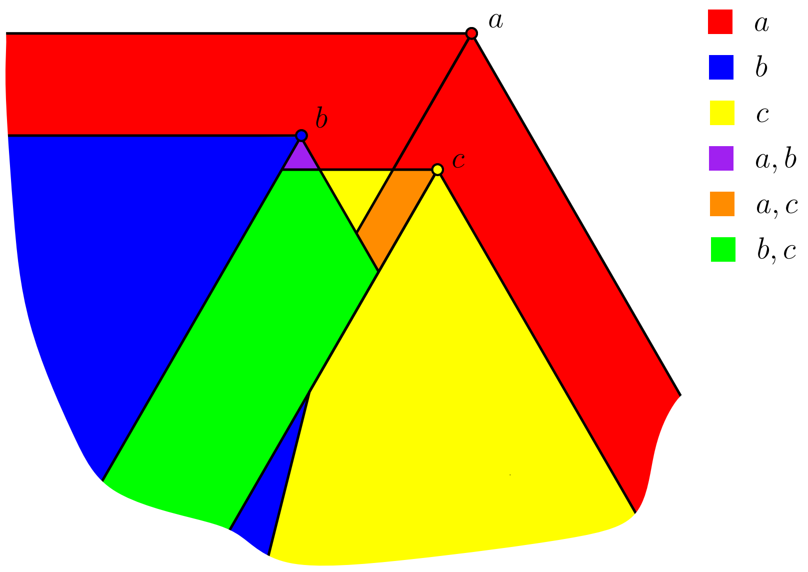


Figure 4.23: An example of an overlay of ODC partitions, showing the overlay of the two partitions in Figure 4.22. The colour in each region indicates the associated terminals. For the general case, with multiple Steiner points and six overlays, each region corresponds to up to six terminals.

ODC regions which intersect to give $R$, and let $N_R = \{t_j : j \in I_R\}$. In other words, $N_R$ is the set of up to six terminals associated with $R$. The importance of the OODC partition lies in the next theorem, which follows directly from Lemma 4.23.

**Theorem 4.26** *Let $R$ be a region of an OODC partition for the terminal set $N$. Let $s$ be any point in $R$. Then there exists a minimum spanning tree $T$ on $N \cup \{s\}$ such that the set of neighbours of $s$ in $T$ is a subset of $N_R$.*

## Updating a minimum spanning tree

To understand the next main step in efficiently constructing a generalised minimum $k$-Steiner tree $T$ on a terminal set $N$, consider first the case where $k = 1$, that is, where there is only a single Steiner point $s$. Since the cost function being minimised is $\ell_1$-optimisable it follows that $T$ is a minimum spanning tree on $N \cup \{s\}$. By Theorem 4.26, there are $O(n^2)$ possibilities for the set of neighbours of $s$ in $T$, namely the subsets of $N_R$, where $R$ ranges over the regions of the OODC of $N$. For a given set of (up to six) neighbours of $s$ we assume that the optimal location of $s$, say $s^*$, can be computed in finite time (an issue which we discuss in more generality at the end of Section 4.3.2). Georgakopoulos and Papadimitriou [173] have shown that there is an $O(n^2)$ preprocessing step that can be performed on the minimum spanning tree of $N$ that will allow it to be updated to a minimum spanning tree on $N \cup \{s^*\}$ in constant time. The tree $T$ can now be constructed by performing this update for every possible $s^*$ and choosing the one with minimum cost.

We now generalise this strategy for $k \geq 1$. In this general case, instead of a single Steiner point $s$ interconnecting a set of terminals, we have a set of up to $k$ Steiner points $S$ forming the interior nodes of a Steiner forest $F$, where the set of leaves of $F$, say $A$, is a subset of $N$. By Lemma 4.23 we only have to consider topologies where nodes have degree at most $6$. Hence $|A| \leq 6k$. In what follows we will show that if we fix the leaf set $A$ and the topology $\mathcal{F}$ of $F$, then with appropriate preprocessing, the minimum spanning tree on $N$ can be updated in constant time to a $k$-Steiner tree on $N$ that is minimum with respect to the given $A$ and $\mathcal{F}$.

To establish this result we first set up some appropriate notation and definitions. Let $\delta_T(s)$ denote the set of neighbours of a node $s$ in a graph $T$. We assume a set of terminals $N$ is given.

**Definition [Viable forest]**: A forest $F$ with node set $A \cup S$, where $A \subseteq N$ and $S \subset \mathbb{R}^2$ with $|S| \leq k$, is called *viable* if and only if $\{t \in V(F) : t \text{ is a leaf of } F\} = A$ and $|\delta_F(s)| \leq 6$ for every $s \in S$.

**Definition [Minimum $F$-fixed spanning tree]**: Let $F$ be a viable forest (with leaf set $A \subseteq N$ and interior nodes $S$). A *minimum $F$-fixed spanning tree* is a tree $T_F$ with minimum cost under the conditions that $V(T_F) = N \cup S$ and $\delta_{T_F}(s) = \delta_F(s)$ for every $s \in S$.

Note that the above definition means that in a minimum $F$-fixed spanning tree $T_F$ the subgraph induced by $A \cup S$ has the same topology as the forest $F$.

In addition to these definitions, we use the notation $P_T(t, y)$ to represent a path through $T$ with endpoints $t$ and $y$, and we use $\mathrm{BSD}_T(t, y)$ to denote the longest edge on $P_T(t, y)$ (or the edge having the bottleneck Steiner distance; see Section 1.3.2 of Chapter 1).

We will make use of the following well-known theorem, which gives an alternative way of recognising minimum spanning trees.

**Theorem 4.27** *[239] A tree $\overline{T}$ is a minimum spanning tree on $N$ if and only if for every $t, y \in N$, $\|e\| \leq \|ty\|$ for every $e \in P_{\overline{T}}(t, y)$.*

We next define a preprocessing procedure required to achieve a constant-time update.

**Definition [Preprocessing procedure PP1]**: Let $\overline{T}$ be a minimum spanning tree on $N$; denote by *PP1* a preprocessing procedure to calculate $\mathrm{BSD}_{\overline{T}}(t, y)$ for every pair of nodes $t, y \in N$.

Clearly, PP1 requires $O(n^2)$ time and $O(n^2)$ space (see Section 1.3.2), and we assume it incorporates a consistent tie-breaking process, based on an ordering of the edges of $\overline{T}$, for choosing between edges of exactly the same length during the procedure.

For the main constant-time update theorem below, we restrict our attention to the case where the forest $F$ is connected (i.e., $F$ is a tree). This is done in order to keep the arguments relatively simple — the case where $F$ contains multiple connected components requires some further preprocessing and is discussed in Section 3.6.2.

**Theorem 4.28** *[52] Given a terminal set $N$ and a constant $k$, let $F$ be a viable tree with leaf set $A \subseteq N$ and interior nodes $S$, with $|S| \leq k$. Let $\overline{T}$ be a minimum spanning tree on $N$, and assume that PP1 has been performed. Then a minimum $F$-fixed spanning tree $T_F$ can be constructed from $\overline{T}$ in $O(k^2)$ time.*

**Proof.** Let $G = \overline{T} \cup F$, and note that $|A| \leq 6k$. A number of cycles may occur in $G$, each one of them containing a path through $F$ with endpoints from $A$. Let $T'$ be the graph obtained by deleting the set of edges $\{\mathrm{BSD}_T(t_i, t_j) : t_i, t_j \in A, i \neq j\}$ from $G$. We will show that $T' = T_F$ for some choice of $T_F$, which suffices to prove the proposition since $T'$ can clearly be constructed in $O(k^2)$ time, due to the preprocessing procedure.

To prove that $T' = T_F$ we first show that $T'$ is acyclic and connected. Every cycle of $G$ is of the form $P_F(t_i, t_j) \cup P_T(t_j, t_i)$ (with $t_i, t_j \in A$), and therefore deleting every $\mathrm{BSD}_T(t_i, t_j)$ from $G$ produces an acyclic graph. We use induction on $|A|$ to prove that $T'$ is connected. Let $A = \{t_1, \ldots, t_{|A|}\}$ and let $A_b = \{t_1, \ldots, t_b\} \subseteq A$ for some $2 \leq b \leq |A|$. Let $F_b$ be the subtree of $F$ induced by $S \cup A_b$, and let $G_b = \overline{T} \cup F_b$. Subtracting $L_b = \{\mathrm{BSD}_T(t_i, t_j) : 1 \leq i < j \leq b\}$ from $E(G_b)$ produces the graph $T_b$. For the base case we let $b = 2$. The only cycle of $G_2$ is $P_F(t_1, t_2) \cup P_T(t_2, t_1)$, and $\mathrm{BSD}_T(t_2, t_1)$ is an edge of this cycle. Therefore, deleting $\mathrm{BSD}_T(t_2, t_1)$ does not destroy the connectivity of $T_2$ on $N \cup S$.

Next suppose, by the inductive assumption, that $T_b$ is connected and acyclic for some $2 \leq b \leq |A| - 1$. This implies there is exactly one path connecting $t_{b+1}$ to a node of $A_b$ not passing through any element of $S$; i.e., this path is of the form $P_T(t_{b+1}, t_r)$ for some unique $t_r \in A_b$. Let $s = \delta_F(t_{b+1})$; that is, $s$ is the Steiner point in $F$ adjacent to $t_{b+1}$. Then $T_{b+1}$ is the graph with $V(T_{b+1}) = N \cup S$ and

$$E(T_{b+1}) = (E(T_b) \cup \{(s, t_{b+1})\}) \setminus \{\mathrm{BSD}_{\overline{T}}(t_{b+1}, t_i) : t_i \in A_b\}.$$

We now make the following claim, the proof of which we leave for Exercise 4.11.

**Claim.** For every $t_i \in A_b$ either $\mathrm{BSD}_T(t_{b+1}, t_i) \in L_b$ or $\|\mathrm{BSD}_T(t_{b+1}, t_i)\| = \|\mathrm{BSD}_T(t_{b+1}, t_r)\|$.

By this claim $E(T_{b+1}) = (E(T_b) \cup \{(s, t_{b+1})\}) \setminus \{\mathrm{BSD}_T(t_{b+1}, t_r)\}$ and we deduce that $T_{b+1}$ can be constructed from $T_b$ by adding one edge from $F$ and then deleting an edge of $T$ on the resultant cycle. This completes the induction argument, and hence $T'$ is connected.

Next we prove that $T'$ is a *minimum* $F$-fixed spanning tree. Let $K$ be the complete graph on $N$. Furthermore, suppose that the edges of $K$ are weighted by the function $w$, where

$$w(t, y) = \begin{cases} 0 & \text{if } t \in A \text{ and } y \in A, \\ \|ty\| & \text{otherwise.} \end{cases}$$

Let $T_A$ be any spanning tree on $A$. Then clearly $T'$ is a minimum $F$-fixed spanning tree if and only if the graph $T_K$, where $V(T_K) = N$ and $E(T_K) = (E(T') \cap (N \times N)) \cup T_A$, is a minimum spanning tree of $K$ with the above weight function. But this follows from a simple application of Theorem 4.27. Hence $T' = T_F$, as required. ∎

### 4.3.2 An algorithm for the generalised $k$-Steiner tree problem

In this subsection we use the properties of generalised $k$-Steiner trees established in Section 4.3.1 to develop an efficient exact algorithm for constructing such trees. We begin by presenting an overall algorithm for solving the generalised $k$-Steiner tree problem, and then outline a crucial sub-procedure that involves generating a minimum $F$-fixed spanning tree for a viable forest $F$. Finally, we discuss some of the practical issues associated with implementing this algorithm.

#### The main algorithm

We begin by presenting the main $O(n^{2k})$ algorithm for solving the generalised $k$-Steiner tree problem in the plane. An outline of the process is given in Algorithm 4.1.

There are four main parts to the algorithm, which we discuss below. Since $k$ is a bounded constant, we ignore factors in the complexity that are functions of $k$.

**Stage 1 - Preliminaries** (Lines 2–5) For the given terminal set $N$ we first construct an OODC partition (in $O(n^2)$ time, by Theorem 4.25), and then a minimum spanning tree $T$ on $N$ (in $O(n \log n)$ time). Preprocessing procedure PP1 is then performed in $O(n^2)$ time. If $k > 1$ a second preprocessing procedure, PP2, is required which is defined as follows.

**Algorithm 4.1:** Generalised minimum $k$-Steiner tree algorithm

```
Input: A set $N$ of $n$ points in a plane with PE norm $\|\cdot\|$, a positive integer $k$, and
       a symmetric $\ell_1$-optimisable function $\alpha$.
Output: A set $S$ of at most $k$ Steiner points, and a tree $T^*$ interconnecting $N \cup S$,
        such that $\|T^*\|_\alpha = \min_{\mathcal{T},S'} \alpha(\mathbf{e}_{\mathcal{T},N,S'})$.
1
2  Construct an OODC partition of $N$
3  Construct a minimum spanning tree $T$ on $N$
4  Perform preprocessing procedure PP1 on $T$
5  if $k > 1$ then Perform preprocessing procedure PP2 on $T$
6
7  // Generate the viable forests
8  foreach choice (with repetition) of $k' \leq k$ regions, $R_1, \ldots, R_{k'}$, of the
   OODC partition do
9      Associate a Steiner point $s_i$ with region $R_i$
10     Construct the graph $\mathcal{G}$ consisting of the vertices $\bigcup N_{R_i} \cup \{s_1, \ldots, s_{k'}\}$, all
       edges $(s_i, s_j), i \neq j$, and all edges $(s_i, t)$ for every $t \in N_{R_i}$
11     Let $\mathcal{G}^*$ be the set of all viable subforests of $\mathcal{G}$
12     foreach $\mathcal{F} \in \mathcal{G}^*$ do
13         Solve the fixed topology generalised Steiner tree problem for $\mathcal{F}$ to get the
           forest $F$
14         Run the minimum $F$-fixed spanning tree algorithm with input $T$ and $F$ and
           let $T_F$ be its output
15 Select a smallest total cost $T_F$ produced and let $T^* = T_F$
16 Let $S$ be the set of Steiner points of $T^*$
```

**Definition [Preprocessing procedure PP2]**: Let $\overline{T}$ be a minimum spanning tree on $N$; denote by *PP2* a preprocessing procedure to calculate a TRUE/FALSE table $H$ such that $H_{e,y,z}$ = TRUE if and only if edge $e \in E(P_T(y, z))$.

Procedure PP2 requires at most $O(n^3)$ time and $O(n^3)$ space; the importance of PP2 will become clear when we discuss the computation of the minimum $F$-fixed spanning trees. In all, this preliminary stage requires $O(n^2)$ time if $k = 1$, or $O(n^3)$ time if $k > 1$.

**Stage 2 - Forest generation** (Lines 8–11) Let $F$ be the subforest of $T^*$ induced by the Steiner points and their neighbours. By Theorem 4.26 the set of leaf neighbours in $F$ of each Steiner point is a subset of the terminals associated with a region of the OODC; since there are at most $k$ Steiner points and $O(n^2)$ regions, the number of such possibilities is $O(n^{2k})$. For each choice of OODC regions in line 8, all possible viable topologies of $F$ are constructed in line 11. The number of such topology choices is a function of $k$ and does not contribute to the complexity of the algorithm.

**Stage 3 - Forest optimisation** (Line 13) This step involves finding a minimum Steiner tree, under the cost function $\alpha$ for $F$ where the topology and leaf nodes are given. As we have seen throughout this book, such computations are generally possible in finite time. Since $|V(F)|$ is assumed to be bounded by a constant (a multiple of $k$), we assume that this step can be performed in constant time. The practicalities of implementing this step will be discussed later in this section.

**Stage 4 - Spanning tree update** (Line 14) If $F$ is a tree (i.e., it has only one connected component), then by Theorem 4.28, $T$ can be updated to incorporate $F$ in constant ($O(k^2)$) time using the constructive method described in the proof of that theorem. A procedure (the minimum $F$-fixed spanning tree algorithm) for generalising this to forests with multiple components will be discussed below. The procedure makes use of PP2, and also runs in constant time ($O(k^{2k+3}k!)$).

If we assume that Stage 4, the spanning tree update, can be successfully completed in constant time, then it is straightforward to see that the generalised minimum $k$-Steiner tree algorithm constructs, in a time of $O(n^{2k})$, a tree $T^*$ that is a generalised minimum $k$-Steiner tree on the terminal set $N$. It remains now to develop an algorithm to construct a minimum $F$-fixed spanning tree for any viable forest $F$.

## The minimum $F$-fixed spanning tree algorithm

Our aim is now to extend Theorem 4.28 to the case where $F$ is not necessarily connected, which must be considered if $k > 1$. We claim that the *minimum $F$-fixed spanning tree algorithm*, given in Algorithm 4.2 and run in line 14 of the generalised minimum $k$-Steiner tree algorithm, computes a minimum $F$-fixed spanning tree for any viable forest $F$. The algorithm makes use of preprocessing procedure PP2 when $k > 1$.

To understand the minimum $F$-fixed spanning tree algorithm, observe first that for $t = 1$ the algorithm is identical to the method outlined in the proof of Theorem 4.28. In other words $D^1 = L^1 = \{\mathrm{BSD}_T(t_i, t_j) : t_i, t_j \in A, i \neq j\}$ and this set is deleted from $G^1 = T \cup F$ to produce $T_F$. The algorithm is iterative in nature, at each step adding a component $F^i$ to the current tree $T^{i-1}$ and then deleting the longest edges on every cycle to get a tree $T^i$. All cycles that are obtained when adding $F^i$ to $T^{i-1}$ are of the form $P_{T^{i-1}}(t, y) \cup P_{F^i}(t, y)$ where $t, y \in A^i$. Similarly to the proof of Theorem 4.28, the required edge to be deleted at Step $i$ for the pair $t, y$, namely $\ell^i(t, y)$, is simply the

**Algorithm 4.2:** Minimum $F$-fixed spanning tree algorithm

```
Input: A set $N$ of $n$ points in the plane, a minimum spanning tree $T$ on $N$, and a
       viable forest $F$ with $\kappa$ connected components $F^1, \dots, F^\kappa$.
Output: A minimum $F$-fixed spanning tree $T_F$ on $N \cup S$.

Let $D^0 = \emptyset$, and let $T^0 = T$

// Iterate over the components of $F$
for $i = 1$ to $\kappa$ do
    Let $A^i = V(F^i) \cap N$
    for every distinct pair $t, y \in A^i$ do
        if $i = 1$ then
            Let $J$ be the graph with $V(J) = \{\{t\}, \{y\}\}$ and $E(J) = \{(\{t\}, \{y\})\}$
            Let $\sigma_1^J((\{t\}, \{y\})) = t$, and let $\sigma_2^J((\{t\}, \{y\})) = y$
        else if $i > 1$ then
            Let $J$ be the graph with $V(J) = \{\{t\}, \{y\}, A^1, \dots, A^{i-1}\}$ and
            $E(J) = \{(U, U') : \exists w \in U \wedge \exists w' \in U'$ such that $H_{e',w,w'} =$
            FALSE $\forall e' \in D^{i-1}\}$
            For every $e = (U, U') \in E(J)$ let $\sigma_1^J(e) = w$ and let $\sigma_2^J(e) = w'$
            (where $w, w'$ are from the previous step)
        Find the unique path in $J$, $P^i = P_J(\{t\}, \{y\})$
        Let $\ell^i(t, y)$ be the edge from $\left\{\mathrm{BSD}_T(\sigma_1^J(e), \sigma_2^J(e)) : e \in E(P^i)\right\}$ of
        maximum length
    Let $L^i = \{\ell^i(t, y) : t, y \in A^i\}$
    Let $D^i = D^{i-1} \cup L^i$
    Let $T^i = (V(T^{i-1} \cup F^i), E(T^{i-1} \cup F^i) \backslash D^i)$
Let $T_F = T^t$
```

longest edge on $P_{T^{i-1}}(t, y)$. It is clear that $P_{T^{i-1}}(t, y)$ is either a path of $T$ or consists of alternating subpaths of $T$ and $F^j$ for various $j < i$. Since $k$ is constant it is possible to find, also in constant time, the subpaths of $P_{T^{i-1}}(t, y)$ that lie in $T$. By taking the maximum of the longest edges of all these paths in $T$ we get $\ell^i(t, y)$.

The purpose of $J$, as defined in the algorithm, is to have a graph of bounded structural complexity that contains a representative edge for every path of $T^{i-1}$ that begins and ends at nodes adjacent to Steiner points and lies wholly in $T$. The specification of the nodes and edges of $J$ in line 12 of the minimum $F$-fixed spanning tree algorithm ensures that $P_{T^{i-1}}(t, y)$ corresponds to a path in $J$ connecting $\{t\}$ and $\{y\}$. Observe that any path $P_{ab}$ in $T^{i-1}$ connecting nodes $t_a \in A^a$ and $t_b \in A^b$ lies entirely in $T$ if and only if $H_{e',t_a,t_b}$ = FALSE for all $e' \in D^{i-1}$, where $D^{i-1}$ is the set of edges that have been

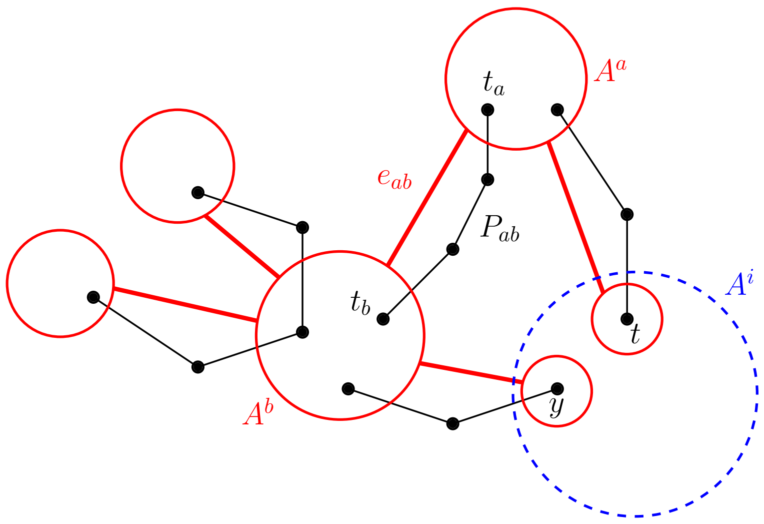


Figure 4.24: An example of graph $J$ at Step $i$. The red circles and lines represent, respectively, the nodes and edges of $J$; the filled black circles and black lines represent some of the nodes and edges of $T$. The path $P_{ab}$ in $T$ corresponds to the edge $e_{ab}$ in $J$ and we have $\sigma_1^J(e_{ab}) = t_a$ and $\sigma_2^J(e_{ab}) = t_b$.

deleted from $T$ up to and including Step $i-1$. Given an edge $e$ of $J$ we need to know the endpoints of the path in $T$ represented by $e$. For this we introduce the functions $\sigma_j^J(e)$, $j = 1, 2$ in the minimum $F$-fixed spanning tree algorithm; see Figure 4.24.

It is straightforward to show by induction that each network $T^i$ generated by the algorithm (in line 18) is a tree; see Exercise 4.12. It immediately follows that there is a unique path in each graph $J$ connecting $\{t\}$ and $\{y\}$, as required in line 14 of the algorithm. (As usual, we assume that a path in $J$ contains no repeated vertices.) A consequence of this is the following lemma.

**Lemma 4.29** *Let $t, y \in A^i$ in the minimum $F$-fixed spanning tree algorithm. Then $\ell^i(t, y)$ (as defined in line 15 of the algorithm) is the longest edge of $P_{T^{i-1}}(t, y)$, excluding any edges of $F$.*

**Proof.** By the previous comments, there is a unique path $P^i = P_J(\{t\}, \{y\})$ for the minimum $F$-fixed spanning tree algorithm to find. The required longest edge on this path is the maximum of the longest edges for each subpath of $W(P^i)$ containing edges of $T$ only. The result follows. ∎

We now verify the correctness and establish the complexity of the minimum $F$-fixed spanning tree algorithm. The theorem implies that even in the case when $F$ is disconnected, our update method, as described in the minimum $F$-fixed spanning tree algorithm, produces an optimal $F$-fixed spanning tree in constant time.

**Theorem 4.30** *Let $T$ be a minimum spanning tree on $N$, and assume that preprocessing steps PP1 and PP2 have been performed. If $F$ is a viable forest, then the minimum $F$-fixed spanning tree algorithm correctly produces a minimum $F$-fixed spanning tree $T_F$ in at most $O(k^{2k+3}k!)$ time.*

**Proof.** The proposition is verified by using induction on $t$ (the number of connected components of $F$). Theorem 4.28 proves the base case: $T^1$ is connected, acyclic, and a minimum $F^1$-fixed spanning tree. Similar reasoning is used to prove that each subsequent $T^i$ is connected and acyclic. At each inductive step, Lemma 4.29 guarantees that the minimum $F$-fixed spanning tree alorithm correctly deletes the longest edge (excluding edges of $F$) of any new cycle formed. Let $F_i = \bigcup_{j \leq i} F^j$. To prove minimality of $T^i$ we once again construct (analogously to Theorem 4.28) a weighted complete graph $K$ on $N$ and a tree $T_K$, such that $T^i$ is a minimum $F_i$-fixed spanning tree on $N \cup S$ if and only if $T_K$ is a minimum spanning tree of $K$. Theorem 4.27 then completes the minimality proof.

To verify the time complexity first note that $t \leq k$, meaning that the outer **for** loop at line 5 is repeated $O(k)$ times. Since $|A^i| \leq 6k$ the inner **for** loop at line 7 is repeated $O(k^2)$ times. Constructing the graph $J$, lines 8 to 13, requires at most $O(k^{2k}k!)$ time, and lines 14 and 15 require $O(k)$ time. The complexity of the algorithm follows. ∎

## Implementation issues

The generalised minimum $k$-Steiner tree algorithm (Algorithm 4.1) shows that the generalised $k$-Steiner tree problem in the plane can be solved in $O(n^{2k})$ time. This is a useful result in understanding the complexity bounds on this very general class of problems. However, it is also natural to ask how practical it is to implement such an algorithm for a given cost function.

In practice there are a number of obstacles to efficiently implementing the main algorithm. The most obvious problem is that of the large hidden constants that take effect as $k$ increases. In particular, we have just seen that the minimum $F$-fixed spanning tree algorithm runs in $O(k^{2k+3}k!)$ time, meaning that Stage 4 (line 14) of the generalised minimum $k$-Steiner tree algorithm becomes slow as $k$ increases.

The other practical difficulty lies in Stage 3 of the generalised minimum $k$-Steiner tree algorithm, the construction of fixed topology Steiner trees for each of the components of $F$. In general the efficient (usually linear-time) methods developed in earlier sections of this book for optimising a Steiner tree with a given Steiner topology cannot be applied directly, as we need to consider topologies other than Steiner topologies (because of the restriction on the number of Steiner points). In particular, we may need to consider topologies where some Steiner points have degree greater than 3. Hence, the question is: For a given cost function $\alpha$ and a given PE norm $\|\cdot\|$, can we solve the fixed topology

Steiner tree problem, where the topology corresponds to a minimum Steiner tree for a bounded number of Steiner points?

Although there are only a few instances of $(\alpha, \|\cdot\|)$ for which the fixed topology problem has been shown to be solvable, there are, as far as we know, no instances of $\alpha$ and $\|\cdot\|$ for which it has been demonstrated that the fixed topology problem is impossible to solve (to within a fixed precision in finite time). Note also that for many cost functions and norms there exist numerical methods (for instance, gradient descent methods) that solve the fixed topology problem to any finite degree of accuracy. We now briefly discuss a few cost functions $\alpha$ for which methods for solving the fixed topology Steiner tree problem have been developed, for some norms.

**1.** $\alpha(\mathbf{e}_{\mathcal{T},A,S}) = \sum \|e_i\|$. This case is the standard geometric Steiner tree problem (as discussed in the first three chapters of this book) for a fixed topology. We have seen that for the Euclidean norm the problem has a linear-time solution, by the Hwang-Melzak algorithm (Theorem 1.5), provided that no point has degree larger than $3$. However, in solutions to the $k$-Steiner tree problem degree $4$ Steiner points do occur, but it has been shown that degree $5$ Steiner points do not occur [331]. The geometric properties of degree $4$ Steiner points lead to a simple finite algorithm for solving this fixed topology Steiner tree problem in the Euclidean plane; see Exercise 4.13. Similar results hold for the rectilinear and other fixed orientation norms, by, for example, using the linear programming formulation described in Section 2.4.1.

**2.** $\alpha(\mathbf{e}_{\mathcal{T},A,S}) = \sum \|e_i\|^p,\ p > 0$. This is referred to as the power-$p$ Steiner tree problem for a fixed topology. In the Euclidean plane with $p = 2$, Ganley [161] has shown that the problem can be solved in linear time. Unfortunately, Ganley's methods do not appear to generalise to the rectilinear or other fixed orientation norms, or to other values of $p$.

**3.** $\alpha(\mathbf{e}_{\mathcal{T},A,S}) = \lim\limits_{p\to\infty} \left(\sum \|e_i\|^p\right)^{1/p}$. This is the bottleneck $k$-Steiner tree problem for a fixed topology. Ganley and Salowe [166] have shown that this problem can be solved in quadratic time in the rectilinear plane. In the Euclidean plane (and other general $\ell_p$ planes) the problem is significantly harder, but there exist various numerical algorithms that can calculate a solution to any desired precision; for example algorithms based on nonlinear optimisation techniques have been proposed by Elzinga et al. [150] and Drezner et al. [133], and an even simpler algorithm based on a binary-search application of an algorithm of Sarrafzadeh and Wong [340] is described in [166]. Furthermore, a fully polynomial-time approximation scheme (FPTAS) exists for the problem in the Euclidean plane (see [161]). More recently, Bae et al. [21] have produced an efficient algorithm for solving the bottleneck $k$-Steiner tree problem in the $\ell_p$ plane that outperforms the generalised minimum $k$-Steiner tree algorithm; this work is discussed in the next section.

## 4.3.3 Bottleneck Steiner trees for the Euclidean and other metrics

In this subsection we develop an approach for solving the bottleneck $k$-Steiner tree problem which is more efficient than the general algorithm in the previous subsection. This approach is based on the papers of Bae, Lee and Choi [22] and Bae, Choi et al. [21]. We will first show, in some detail, how to solve the bottleneck 1-Steiner tree problem in the Euclidean plane, and then outline how this method can be generalised to larger values of $k$ and other metrics.

### The bottleneck 1-Steiner tree problem in the Euclidean plane

As indicated at the beginning of Section 4.3, we can assume that a solution $T$ to the bottleneck 1-Steiner tree problem for a given set of terminals $N$ is a tree. Furthermore, since the bottleneck cost function is an example of an $\ell_1$-optimisable function, it follows that $T$ can be assumed to be a minimum spanning tree on its nodes (which comprise the elements of $N$ and at most one extra node). The idea is to show that $T$ can be constructed by modifying any minimum spanning tree of $N$. The key to achieving this lies in the following two lemmas.

The first lemma is a simple application of Theorem 4.27 (Exercise 4.14).

**Lemma 4.31** *Let $\overline{T}$ be a minimum spanning tree of $N$. Then there exists a minimum bottleneck 1-Steiner tree $T$ for $N$ with Steiner point $s$ such that each edge of $T$ is either an edge of $\overline{T}$ or is incident to $s$.*

**Lemma 4.32** *Let $\overline{T}$ be a minimum spanning tree of $N$, and let $e_1, \ldots, e_{n-1}$ be the edges of $\overline{T}$ listed in non-increasing order of edge length. Then, either $\overline{T}$ is a minimum bottleneck 1-Steiner tree for $N$ or there exists a minimum bottleneck 1-Steiner tree $T$ for $N$ with Steiner point $s$ of degree $c + 1 > 1$ such that the edges of $T$ not incident with $s$ are $e_{c+1}, \ldots, e_{n-1}$.*

**Proof.** If $\overline{T}$ is not a minimum bottleneck 1-Steiner tree for $N$, then we know that there exists such a tree $T$ that satisfies the conditions of Lemma 4.31, that is, all edges not incident to the Steiner point belong to $\overline{T}$. Suppose we choose this tree $T$ so that the degree of the Steiner point $s$ is as small as possible.

Let $b$ be the bottleneck length of $T$. We claim that $b < |e|$ , where $e$ is the shortest edge in $\overline{T}$ not in $T$. If the claim is not true, then adding $e$ to $T$ does not increase the bottleneck length but creates a cycle which includes the Steiner point. So, by then deleting one of the edges incident to $s$ we create another tree with bottleneck length no greater than $b$ and satisfying the conditions of Lemma 4.31, but where $s$ has smaller degree. Hence,

**Algorithm 4.3:** Bottleneck 1-Steiner algorithm

```
Input:  A set $N$ of $n$ points in the plane.
Output: A minimum bottleneck 1-Steiner tree $T$ for $N$; $b$, the bottleneck length
        of $T$.

Construct $T_0$, a minimum spanning tree of $N$
Let $e_1, \dots, e_{n-1}$ be the edges of $T_0$ listed in non-increasing order of edge length
Let $S = \emptyset$, let $b = |e_1|$

// Iterate over the possibilities for the degree ($c + 1$) of the Steiner point
for $c = 1$ to $4$ do
    Let $T_c = T_{c-1} \setminus \{e_c\}$
    Construct the smallest radius disc $D_c$ containing a node from each connected
    component of $T_C$
    Let $s_c$ be the centre of $D_c$, and let $r_c$ be the radius of $D_c$
    if $r_c < b$ then Set $S = \{s_c\}$ and $b = \max\{r_c, |e_{c+1}|\}$
Let $T$ be a minimum spanning tree of $N \cup S$
```

by contradiction, the claim is true. The statement of the lemma immediately follows from the claim. ∎

It is well known, and easy to prove, that no node in a Euclidean minimum spanning tree can have degree greater than $6$, so $6$ is an upper bound for $c$ in the above lemma. In fact, it has been shown by Monma and Suri [288] that for any set of nodes there exists a Euclidean minimum spanning tree with maximum degree $5$. It follows from their arguments that we can restrict the value of $c$ in Lemma 4.32 to at most $5$.

Lemma 4.32 suggests a straightforward way to solve the bottleneck 1-Steiner tree problem, outlined in Algorithm 4.3. Analysing the computational complexity of Algorithm 4.3 requires an understanding of how efficiently line 9 of the algorithm can be performed. Suppose, once we remove the $c$ longest edges of $T_0$, we assign colours to the terminals according to which of the $c + 1$ connected components of the resulting forest they belong to. See Figure 4.25. Then the disc $D_c$ in line 9 is known as the *smallest colour spanning disc*, that is, the disc of smallest radius containing at least one terminal of each colour. A brute force method for constructing $D_c$ is to consider all possible combinations obtained by selecting one terminal of each colour and construct the smallest enclosing disc for each combination; this clearly requires $O(n^{c+1})$ time. A much more efficient approach makes use of the farthest colour Voronoi diagram (see [2]), defined below.

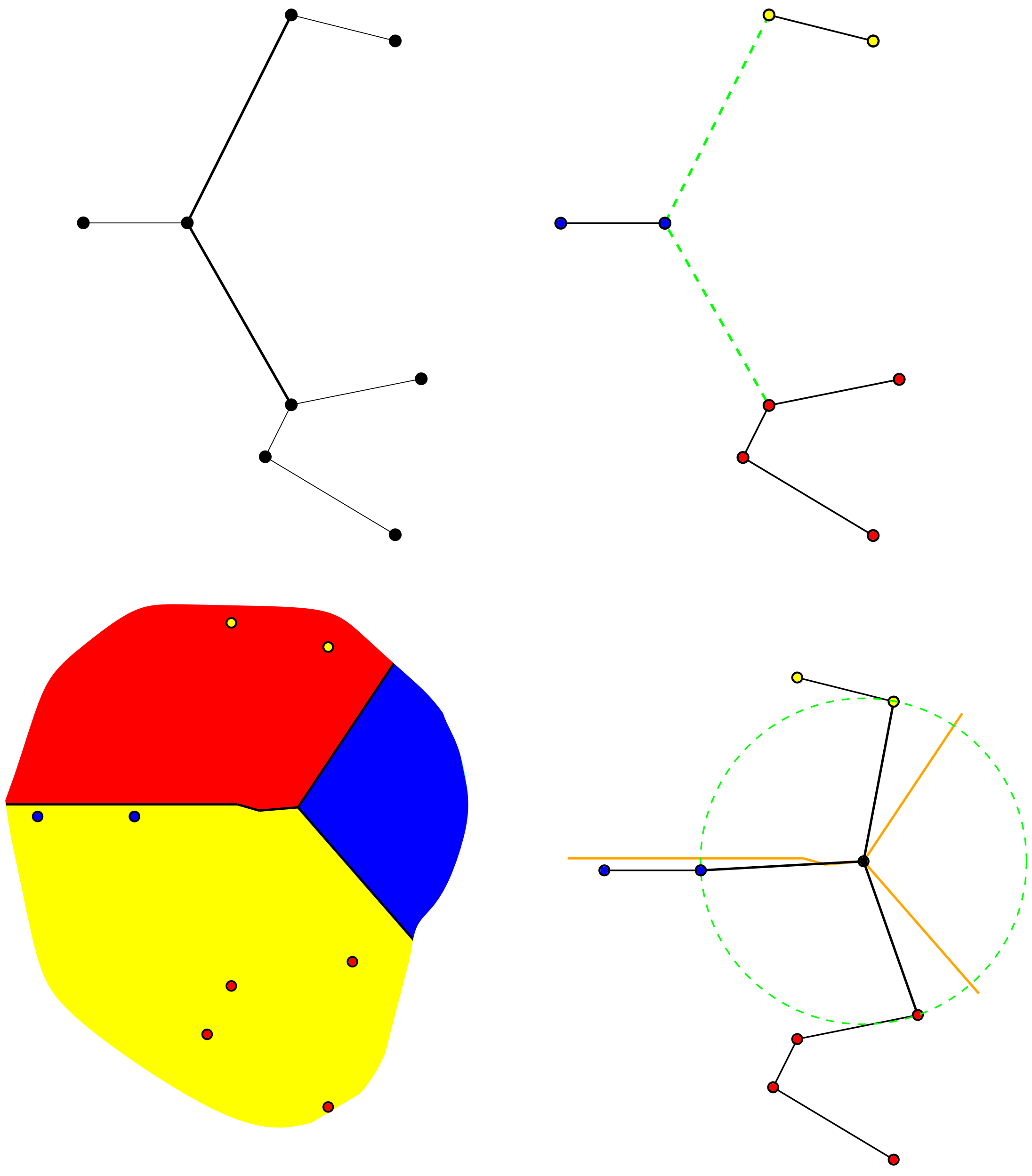


Figure 4.25: Construction of a Euclidean bottleneck 1-Steiner tree for eight terminals: First a minimum spanning tree on the terminals is constructed, and then the longest $c$ edges (where here $c = 2$) are deleted. The terminals are assigned colours depending on which connected component they belong to, and then the farthest colour Voronoi diagram for the coloured terminals is constructed. Finally the Voronoi diagram is used to locate the centre of the smallest colour-spanning disc; this centre forms the location of the Steiner point which reconnects the three connected components.

**Definition [Farthest colour Voronoi diagram]**: Given a collection $C = \{P_1, \ldots, P_{c+1}\}$ of $c+1$ sets of coloured points and a point $x \in \mathbb{R}^2$, define the distance of $x$ to a colour $i$ (for $1 \leq i \leq c+1$) as $d_i(x) := \min_{p \in P_i} |xp|$. Then, the *farthest colour Voronoi diagram*, FCVD($C$), is a decomposition of $\mathbb{R}^2$ into Voronoi regions $VR_i$ for each subset $P_i$, where $VR_i = \{x \in \mathbb{R}^2 | d_i(x) > d_j(x), 1 \leq j \leq c+1, j \neq i\}$.

An example of a farthest colour Voronoi diagram is shown in Figure 4.25 (lower left diagram), where the colour of each Voronoi region corresponds to the colour of the points from which it is most distant. Note that each edge of FCVD($C$) is the set of points that have the same set of two equally farthest colours, while each vertex of FCVD($C$) is a point that has three or more farthest colours.

It follows that the centre of a smallest colour spanning disc is either a vertex of the corresponding farthest colour Voronoi diagram, or, in degenerate cases, the midpoint of two terminals of different colours. Using this characterisation, Abellanas et al. [2] have given a straightforward algorithm for constructing a smallest colour spanning disc in $O(c^3 n \log n)$ time. Since a minimum spanning tree on $n$ points can be constructed in time $O(n \log n)$, we get the following result.

**Theorem 4.33** *Let $N$ be a set of $n$ terminals in the Euclidean plane. Then the bottleneck $1$-Steiner algorithm (Algorithm 4.3) correctly produces a minimum bottleneck $1$-Steiner tree $T$ for $N$ in at most $O(n \log n)$ time.*

**Proof.** The correctness of the algorithm follows from Lemma 4.32; the time complexity follows from the discussion above. ∎

### The general bottleneck $k$-Steiner tree problem

We now briefly outline how the above approach can be generalised to $k > 1$ Steiner points, and general $\ell_p$ metrics. A complete treatment of the problem is given in [21] and involves numerous technical details, which we omit here. The key properties to developing an efficient exact algorithm are those given in the following theorem.

**Theorem 4.34** *Let $N$ be a set of $n$ terminals in the $\ell_p$ metric plane. Let $\overline{T}$ be a minimum spanning tree of $N$, and let $e_1, \ldots, e_{n-1}$ be the edges of $\overline{T}$ listed in non-increasing order of edge length. Then for any given integer $k$ there exists a bottleneck $k$-Steiner tree $T$ that satisfies the following conditions.*

1. *Each edge in $T$ is either an edge of $\overline{T}$ or is incident to a Steiner point.*

2. *Each Steiner point is located at the centre of the minimum enclosing $\ell_p$ circle of its neighbours in $T$.*

3. *Each Steiner point of $T$ has degree at most $5$.*

4. *There is a positive integer $c$ with $1 \leq c \leq 4k$ such that $T$ excludes $e_1, \ldots, e_c$ but includes $e_{c+1}, \ldots, e_{n-1}$.*

For details of the proof, see [21].[51] In brief, conditions 1–3 follow from arguments similar to those given above for the bottleneck 1-Steiner tree problem in the Euclidean plane. For condition 4, note that the bounds on the number of Steiner points (namely $k$) and on the degree of each Steiner point ($5$) imply that if we delete all Steiner points and their incident edges from $T$, then the resulting number of connected components is at most $4k + 1$ (at most 5 components for the first Steiner point to be removed, and up to an extra 4 components for each subsequent Steiner point); condition 4 immediately follows.

Using the properties in Theorem 4.34 the strategy for solving the bottleneck $k$-Steiner tree problem is similar to that outlined in Algorithm 4.3. We first construct a minimum spanning tree $\overline{T}$ for $N$, then for each $c$ satisfying $1 \leq c \leq 4k$ we remove from $\overline{T}$ the $c$ longest edges $e_1, \ldots, e_c$. This results in $c + 1$ induced disjoint subtrees $T_1, \ldots, T_{c+1}$. We then need to find up to $k$ Steiner points to reconnect the $T_i$'s with new edges incident to the Steiner points, minimising the longest edge length. These new edges form a Steiner forest $F$ where all internal nodes are Steiner points and all leaf nodes are elements of $N$. Finally we choose the best solution amongst all $c$ as an optimal solution for the original problem.

The main challenge remaining is that of constructing a suitable Steiner forest $F$ minimising the bottleneck length. The approach given in [21] is to first list all possible topologies for the Steiner forest $F$, where the leaves are labelled only by their associated connected component $T_i$ rather than the precise terminal — the number of such topologies is bounded by $O((4k + 1)^{4k-1})$. For each topology we enumerate all possible so-called 'determinator lists'; these are essentially partial orderings on the edges of a given topology for $F$ that determine the relative lengths of edges in the forest. For each topology it can be shown that there are at most $2^{O(k)} n^k$ distinct determinator lists that need to be considered, and that they can be enumerated in the same time bound. This result makes use of the properties of farthest colour Voronoi diagrams (as discussed previously for the Euclidean $k = 1$ case). Each determinator list can then be used to compute a minimum associated bottleneck length and the locations of the Steiner points in time $O(2^{O(k)}) + O(k^2 \log k)$.

Together, these bounds result in the following theorem.

**Theorem 4.35** *[21] Given $n$ terminals in the plane and a positive integer $k$, a bottleneck $k$-Steiner tree in the $\ell_p$ metric, for any fixed rational $p$ with $1 < p < \infty$, can be computed in $f(k) \cdot (n^k + n \log n)$ time, where $f(k) = k^{5k} \cdot 2^{O(k)}$.*

Finally, we note that when $k = 1$, the proof of Theorem 4.33 can easily be adapted to solve the bottleneck 1-Steiner tree problem for any $\ell_p$ metric in time $O(n \log n)$. It is also shown in [21] that for the $\ell_1$ and $\ell_\infty$ metrics the bottleneck $k$-Steiner tree problem has the remarkable property of being fixed-parameter tractable, as it can be solved in time $O(f(k) \cdot n \log^2 n)$, for some suitable function $f$.

## 4.3.4 Applications

The most important practical application of efficient algorithms for the $k$-Steiner tree problems discussed above is to the design of wireless sensor networks. They are particularly relevant to the problems of relay deployment and augmentation. They have also been shown to be useful in the development of effective heuristics for general Steiner tree problems. We discuss both these applications below.

### Wireless sensor networks

Wireless sensor networks (WSNs) consist of small low-powered sensing and relay devices that can be readily deployed in diverse environments to form distributed wireless networks for collecting information in a robust and autonomous manner. A schematic illustration of a multi-hop WSN allowing data to be sent from a set of sensors to a central base station is given in Figure 4.26. Although early research was mainly motivated by potential military uses, there are now many other important applications such as fault detection, natural disaster early warning systems, and health care and environmental monitoring (see, for example, [300]). An example of the latter is the recent deployment of wireless sensors to monitor factors that contribute to coral bleaching in the Great Barrier Reef in Australia [23]. Other applications include pollution control, climate control in large buildings, and medical applications using implantable devices.

The theory of bottleneck and other $k$-Steiner trees is particularly relevant to the design and layout of WSNs, especially with regard to the locations of relays. Suppose we assume that an initial WSN deployment phase has already taken place and that the locations of the sensors are known. For many applications the locations of the sensors will be determined by the positions of specific data sources, and hence can be assumed to be fixed. Furthermore, we can generally assume that the base station, where all the field data is collected and processed, also has a fixed (or at least highly constrained) location. In the next design phase, known as the *relay augmentation phase*, a bounded number of relays are deterministically introduced to the region, resulting in a multi-hop routing tree allowing data from the sensors to be passed to the base station, and similarly allowing instructions from the base station to be communicated back to the sensors. The relays can be thought of as Steiner points of the network and ideally they should be located so

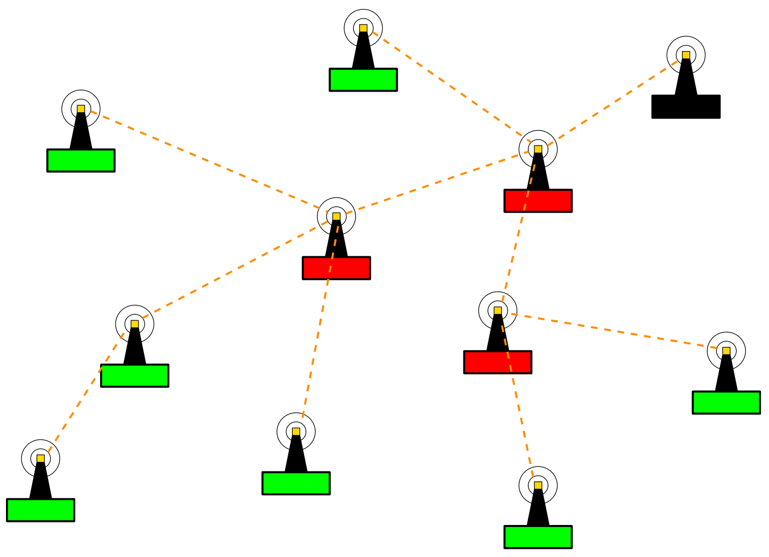


Figure 4.26: A schematic diagram of a multi-hop wireless sensor network. The sensors are indicated in green, the relays in red, and the base station in black. The network transmits data from the sensors to the base station (for collection and analysis) via the intermediary relays. The dashed orange lines show the routing network.

as to optimise some objective function, such as minimising the cost or maximising the performance of the WSN. The number of relays is assumed to be bounded due to the relatively high cost at present of reliable sensors and relays.

Two typical objectives in the relay augmentation phase are to either: (1) maximise the lifetime of the network; or (2) minimise the total energy consumption of the network. Achieving either of these objectives requires an understanding of how the locations of the relays influence power consumption. The idea is that once the locations of the relays have been determined a routing tree rooted at the base station and spanning all nodes can be constructed (as illustrated by the dashed orange lines in Figure 4.26). The power consumption at each sensor and relay is dominated by the power required to transmit data to its parent in the rooted tree. At this stage the power level of each sensor and relay can be adjusted so that each node is able to transmit data to its parent in the routing tree. (We assume, in the simplest models, that all sensors and relays transmit data at the same constant rate throughout the lifetime of the network.) The energy consumed during data transmission is an increasing function of the communication distance; sometimes this function is modelled as simply being proportional to the distance, but in more realistic models it may be taken to be a multiple of some power (usually between 2 and 4) of the communication distance [385].

For objective (1) above, assume that each relay and sensor has an equal finite power supply (such as a battery). If we measure the lifetime of the network as the time until the first sensor or relay exhausts its power supply, then the problem of maximising this lifetime corresponds to minimising the length of the longest link in the network, and hence is equivalent to solving the bottleneck $k$-Steiner tree problem (where $k$ is the number of relays). For objective (2), assume that through regular maintenance the batteries for each sensor or relay are recharged or replaced as needed. Then minimising the energy consumption of the network is equivalent to minimising the sum of energy costs across all the edges of the routing tree. Hence, the objective is equivalent to solving a form of the generalised $k$-Steiner tree problem.

It follows that the algorithms developed in this section form a fundamental foundation for solving the relay augmentation problem for WSNs. However, it is important to also be aware of the limitations of these exact algorithm. One key issue is the difficulty in implementing these algorithms, as discussed at the end of Section 4.3.2; even for small values of $k$ the algorithms contain large hidden constants which may make implementation impractical unless one is willing to compromise optimality. Another issue is that the models described above are simplifications of reality that do not take into consideration some of the costs and constraints that are vital to good layout design for the network.

For example, some WSNs require a *robustness constraint*, which guarantees that the unexpected failure of a single node (or a given number of nodes) does not kill off the rest of the network. This constraint can be modelled by requiring that the routing network is a *$\mu$-connected network* rather than a tree. By a $\mu$-connected network we mean a network that remains connected when any $\mu - 1$ nodes and their incident edges are removed; see Figure 4.27. Even for $\mu = 2$ this added constraint makes the generalised $k$-Steiner tree problem substantially more difficult. This is not surprising, given that for $k = 0$ (that is, for the spanning network version) the problem is already NP-hard [32]. However, it has recently been shown that for the 2-connected bottleneck $k$-Steiner problem efficient polynomial-time algorithms exist that build upon the methods of Bae et al. [21]; for details see the papers of Brazil et al. [54, 56].

## Heuristics

The other significant application of exact algorithms for the $k$-Steiner tree problems has been in the development of heuristics. Although the algorithms for the generalised $k$-Steiner tree problems present real challenges in terms of implementation, for some metrics it is possible to find efficient implementations for the case where $k = 1$, which can be used as the basis of a heuristic method for more difficult problems.

The earliest example of this was the Iterated 1-Steiner algorithm of Khang and Robins [230]. This algorithm is a heuristic for solving the rectilinear Steiner tree problem (of Chapter 3). Starting with the given set of $n$ terminals, the algorithm simply solves the rectilinear 1-Steiner problem in $O(n^2)$ time (using a variant of the method developed

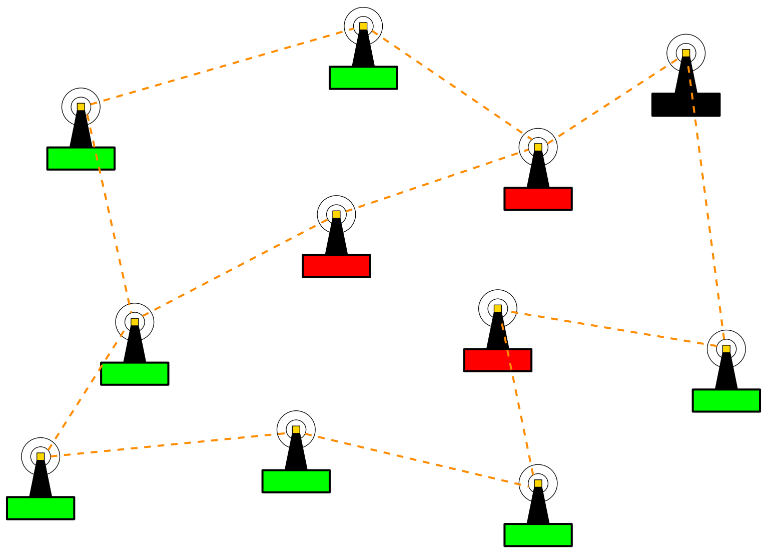


Figure 4.27: A 2-connected routing network for the sensors, relays and base station from the example in Figure 4.26. Note that the network remains connected if any single sensor or relay fails.

by Georgakopoulos and Papadimitriou [173]), adds this Steiner point to the set of fixed nodes, and repeats the procedure $n$ times. This heuristic clearly runs in time $O(n^3)$, and has been shown to be a very effective approach. At the time it was introduced, in the early 1990s, it outperformed all other known heuristics, most of which were based on making local improvements to the rectilinear minimum spanning tree. Subsequently, however, equally effective but computationally faster methods have been developed, such as the edge-based heuristic of Borah et al. [37].

Along similar lines, a heuristic algorithm for the Euclidean bottleneck $k$-Steiner tree problem has been developed (in [55]) based on iteratively solving the Euclidean bottleneck 1-Steiner tree problem, using the exact $O(n \log n)$-time algorithm of Theorem 4.33. Again, this has been shown to outperform all previous known heuristics for this problem.

## 4.4 Trees minimising flow costs

The aim of an interconnection network, in an engineering context, is generally to facilitate the movement of traffic between a number of sites. The specific type of traffic can vary from one application to the next: from cars, to lightwaves, to electricity, to digital signals; all of these different types of traffic can be modelled as flows on the networks. In this section we look at the effect of including costs associated with these flows as part of the overall cost of the network, and how we design optimal networks that minimise such

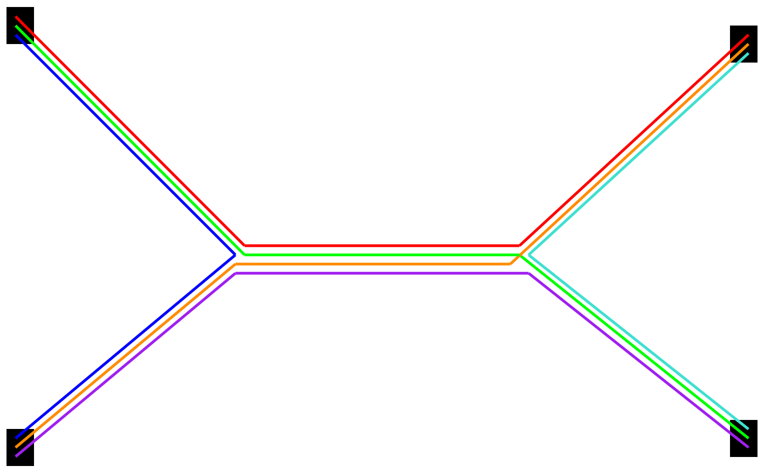


Figure 4.28: A representation of a Gilbert network on four terminals with two Steiner points. For each pair of terminals (indicated by the black rectangles) there is a given symmetric flow (indicated by the different coloured paths between pairs of terminals). Here the topology of the network determines the flows associated with each edge. The weight of each edge is a function of the total flows through that edge.

costs. As usual we will focus on a geometric version of this problem, where the network can contain extra nodes that can be located anywhere in the plane.

### 4.4.1 Gilbert networks and arborescences

The idea of incorporating flow into the Steiner tree problem dates back to a paper of Gilbert [179] in 1967.[52] Gilbert proposed a generalisation of the Euclidean Steiner tree problem by incorporating into the network cost a set of flows between terminals. In this section we consider this flow version of the Steiner tree problem, and its extension to networks in general Minkowski spaces. Much of the discussion here is based on [379].

#### Preliminaries

In Gilbert's generalisation of the Steiner tree problem, symmetric non-negative flows are assigned between each pair of terminals. The aim is to find a least-cost network interconnecting the terminals, where each edge has an associated total flow such that the flow conditions between terminals are satisfied and there is conservation of flow at each Steiner point (see Figure 4.28).

The cost of an edge is its length multiplied by a non-negative weight, where the weight is determined by a given function of the total flow being routed through that edge.

This weight function should satisfy a number of conditions which we discuss below. Originally Gilbert networks were defined only for Euclidean space, but here we extend the definition to general Minkowski spaces.

Let $T$ be a network interconnecting a set $N = \{t_1, \ldots, t_n\}$ of $n$ terminals in a Minkowski space. For each pair $t_i, t_j,\ i \neq j$, of terminals, a non-negative *flow* $f_{ij} = f_{ji}$ is given. The cost of an edge $e$ in $T$ is $w(f_e)l_e$, where $l_e = \|e\|$ is the length of $e$ under the given norm, $f_e$ is the total flow being routed through $e$, and $w(\cdot)$ is a unit cost *weight function* defined on $[0, \infty)$ satisfying

(W1) $\quad w(0) \geq 0 \quad$ and $\quad w(f) > 0$ for all $f > 0$,

(W2) $\quad w(f_2) \geq w(f_1) \quad$ for all $\quad f_2 > f_1 \geq 0$,

(W3a) $\quad w(\cdot)$ is a concave function.

That the function $w$ is concave means by definition that $-w$ is convex. Conditions (W1) and (W3a) imply the following subadditivity condition:

(W3b) $\quad w(f_1 + f_2) \leq w(f_1) + w(f_2) \quad$ for all $\quad f_1, f_2 > 0$.

**Definitions [Gilbert network, weight of an edge]**: Let $w$ be a weight function satisfying conditions (W1), (W2) and (W3b) above. A *Gilbert network* is a network interconnecting a given set of terminals, with given symmetric non-negative flows between each pair of terminals, such that the cost of each edge $e$ is a product of its length $l_e$ and its weight $w(f_e)$, where $f_e$ is the total flow through $e$.

The *total cost* of a Gilbert network $T = (V(T), E(T))$ is the sum of all edge costs, that is,

$$C(T) \;=\; \sum_{e \in E(T)} w(f_e)l_e.$$

GILBERT NETWORK PROBLEM IN THE PLANE
**Given**: A set of points $N$ lying in the plane, a norm $\|\cdot\|$, a symmetric non-negative flow demand between each pair of terminals in $N$, and a weight function $w$ satisfying conditions (W1), (W2) and (W3b) above.
**Find**: A Gilbert network $T = (V(T), E(T))$, for $N$ and the given flow demands, whose cost $C(T)$ with respect to the given weight function is minimum amongst all such Gilbert networks.

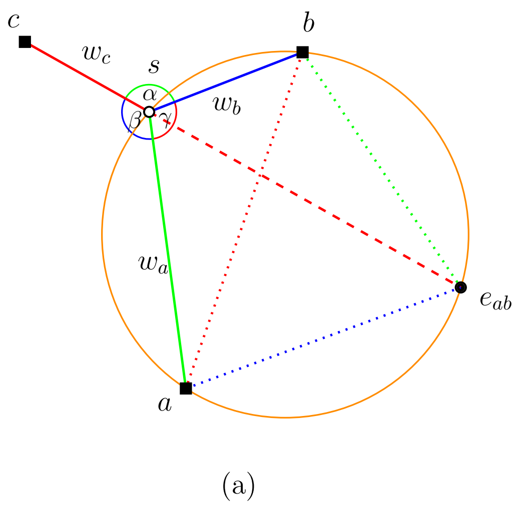
A network $T$ solving the Gilbert network problem in the plane is called a *minimum Gilbert network*. A straightforward argument of [124] (based on establishing a bound on the number of Steiner points) shows that a minimum Gilbert network always exists in a Minkowski space when conditions (W1), (W2) and (W3b) are assumed for the weight function. These three conditions are very natural conditions for most applications; however, they are not enough to guarantee that a minimum Gilbert network is a tree, as we will show later in this section.

The assumption, in the definition of the Gilbert network problem, that the flows are symmetric, while generally reasonable in the design of communications networks, is not a valid assumption for many other applications. An important variation on the Gilbert network problem, that we will show to be a special case of the Gilbert network problem, occurs when the terminals consist of $n-1$ sources and a single sink (or one source and $n-1$ sinks) and all flows not between a source and a sink are zero. The general cost setup here is the same as for the Gilbert network problem, except that most edges now have fewer flows being routed through them, and these flows are now directed. For example, in Figure 4.28 if the terminal in the top left-hand corner is the sink, then the network contains the red, green and dark blue flows only, and all flows are oriented towards the sink.

If the weight function in the single sink problem satisfies the four conditions (W1), (W2), (W3a), (W3b), then it follows from [179] that the resulting minimum network has a tree topology, and provides a directed path from each source to the sink. In this case the weight associated with each edge is uniquely determined by the tree topology and the flow demands at the terminals. Such a network is an arborescence (a rooted tree), and allows us to define the following problem.

GILBERT ARBORESCENCE PROBLEM IN THE PLANE
**Given**: A set of points $N$ lying in the plane consisting of $n$ sources and a single sink, a norm $\|\cdot\|$, a directed flow demand between each source and the sink, and a weight function $w$ satisfying conditions (W1), (W2), (W3a), (W3b) above.
**Find**: An arborescence $T$, for $N$ and the given flow demands, whose cost $C(T)$ with respect to the given weight function is minimum amongst all such arborescences.

An arborescence $T$ solving the above problem is called a *minimum Gilbert arborescence*. Traditionally, the term ‘arborescence’ has been used to describe a rooted tree providing directed paths from the unique root (source) to a given set of sinks. Here, based on a number of applications which we describe in Section 4.4.2, we consider the case where the flow directions are reversed; that is, the flow is from $n$ sources to a unique sink. It is clear, however, that the resulting weights for the two problems are equivalent, hence



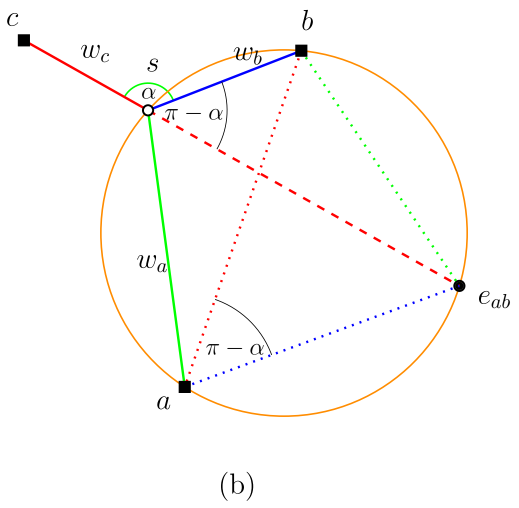


Figure 4.29: A Steiner point $s$, and the construction of the weighted equilateral point $e_{ab}$.

we will continue to use the term ‘arborescence’ for the latter case. Moreover, if we take the sum of these two cases, and rescale the flows (dividing flows in each direction by 2), then again the weights for the total flow on each edge are the same as in the previous two cases, and the flows are symmetric. This justifies our earlier claim that the Gilbert arborescence problem can be treated as a special case of the Gilbert network problem. It will be convenient for the remainder of this section to think of an arborescence as a network with a unique sink.

### Constructing Steiner points, and the importance of concavity

We now describe a method for geometrically constructing a degree $3$ Steiner point in a Euclidean minimum Gilbert network, assuming that the weights of the three incident edges are known. We will then use this method to show that, without the concavity property, condition (W3a), we cannot guarantee that a minimum Gilbert network is a tree.

Let $s$ be a degree $3$ Steiner point of a minimum Gilbert network, with adjacent vertices $a, b$ and $c$. Let the weights in $T$ of the three edges incident with $s$ be $w_a$, $w_b$ and $w_c$, respectively (as in Figure 4.29(a)). Note, by conditions (W2) and (W3b), that there exists a triangle with side lengths $w_a$, $w_b$ and $w_c$; we call this triangle the *weight triangle* of $s$. Let $\alpha = \angle bsc$, $\beta = \angle asc$ and $\gamma = \angle asb$ be the three meeting angles at $s$. Similarly to the unweighted Euclidean case (in Chapter 1), the relative minimality of $T$ implies that the sum of the three weighted unit vectors $w_a\widehat{sa}$, $w_b\widehat{sb}$ and $w_c\widehat{sc}$ must equal $0$. Hence, it is straightforward to show (Exercise 4.15) that these angles satisfy

$$\cos\alpha = \frac{w_a^2 - w_b^2 - w_c^2}{2w_b w_c} \tag{4.3}$$

$$\cos\beta = \frac{w_b^2 - w_a^2 - w_c^2}{2w_a w_c} \tag{4.4}$$

$$\cos\gamma = \frac{w_c^2 - w_a^2 - w_b^2}{2w_a w_b} \tag{4.5}$$

and furthermore that the angles of the weight triangle of $s$ are $\pi - \alpha$, $\pi - \beta$ and $\pi - \gamma$. These results imply that if the edge weights and topology of a minimum Gilbert network are known, then the angles $\alpha$, $\beta$, $\gamma$ around a degree $3$ Steiner point are entirely determined by the weights $w_a$, $w_b$, $w_c$ on the incident edges, and hence are independent of the locations of the neighbouring vertices.

We now describe a method for constructing $s$ that results in a recursive algorithm, similar to Melzak's algorithm for Euclidean Steiner trees.[53] Let $C$ be the circumcircle of triangle $\triangle asb$ (indicated in orange in Figure 4.29), let $L_{cs}$ be the line extending $cs$, and let $e_{ab}$ be the intersection of $L_{cs}$ and $C$, other than $s$. These constructions are illustrated in Figure 4.29(a). Note that if we move the vertex $c$, while fixing $a$ and $b$, then $s$ moves along the arc of $C$ between $a$ and $b$ since the angle $\gamma$ is determined only by the weights and not the location of $c$. Furthermore, we have the following lemma.

**Lemma 4.36** *The location of the point $e_{ab}$, as described above, is independent of the location of $c$ .*

**Proof.** Referring to Figure 4.29(b), note that $\angle bae_{ab} = \pi - \alpha$ since the angles $\angle bse_{ab}$ and $\angle bae_{ab}$ are both subtended by the same arc of $C$ and hence are equal. Similarly, $\angle abe_{ab} = \pi - \beta$. It follows that the location of $e_{ab}$ depends only on the locations of $a$ and $b$ and the weights $w_a$, $w_b$ and $w_c$. ∎

It follows from the above proof that $\triangle abe_{ab}$ is similar to the weight triangle. The point $e_{ab}$ is called a *weighted equilateral point*, in analogy with the unweighted case where $e_{ab}$ is the third point of an equilateral triangle. In a sense, $\triangle abe_{ab}$ is a weighted equilateral triangle since $|ab|/w_c = |ae_{ab}|/w_b = |be_{ab}|/w_a$. Hence the term for $e_{ab}$ and its role are both consistent with the unweighted equilateral point from Chapter 1.

Given points $a$, $b$ and $c$, and weights $w_a$, $w_b$ and $w_c$ we can now construct $e_{ab}$ and $s$ as follows. Compute $\alpha, \beta$ from Equations (4.3), (4.4); then $e_{ab}$ is the intersection of two rays originating at $a$ and $b$ making respective angles $\pi - \alpha$ and $\pi - \beta$ with $ab$, such that the intersection is on the opposite side of $ab$ to $c$. Finally, $s$ is the intersection of the interior of the line segment $ce_{ab}$ with the circumcircle of $\triangle abe_{ab}$.

The line $ce_{ab}$ is called a *generalised Simpson line* [224], and has the property stated in the theorem below, analogous to the Simpson line in the unweighted case. The proof of this theorem uses Ptolemy's theorem, which states that in a cyclic quadrilateral, the product of the diagonals is equal to the sum of the products of the two pairs of opposite sides.[54]

**Theorem 4.37** *Let $s$ be a degree $3$ Steiner point of a minimum Gilbert network, with adjacent vertices $a, b$ and $c$ and corresponding weights $w_a$, $w_b$ and $w_c$. Let $e_{ab}$ be the weighted equilateral point, as described above. Then $w_c|ce_{ab}| = w_a|as| + w_b|bs| + w_c|cs|$.*

**Proof.** First note that $asbe_{ab}$ forms a cyclic quadrilateral; see Figure 4.29. By Ptolemy's theorem we have $|ab||e_{ab}s| = |as||be_{ab}| + |ae_{ab}||bs|$, and hence

$$|e_{ab}s| = |as|\frac{|be_{ab}|}{|ab|} + |bs|\frac{|ae_{ab}|}{|ab|}.$$

Since $\triangle abe_{ab}$ is similar to the weight triangle for $s$ we obtain

$$|e_{ab}s| = |as|\frac{w_a}{w_c} + |bs|\frac{w_b}{w_c};$$

therefore, $w_c|e_{ab}s| = w_a|as| + w_b|bs|$, and the result follows by adding $w_c|cs|$ to each side of the equation. ∎

It follows from the above results that if a minimum Gilbert network has a known tree topology and all Steiner points have degree $3$, then we can construct the network using Melzak's algorithm from Chapter 1 (that is, building it up recursively from the cherries). Furthermore, the weighted length of the resulting generalised Simpson line will equal the cost of the network. However, for this weighted problem there is no obvious way of solving the side problem, that is, for any two terminals $a$ and $b$ adjacent to a Steiner point determining the side of $ab$ on which $e_{ab}$ should be constructed. This means that although we have a finite algorithm for constructing such a Gilbert network, we cannot guarantee that it will run in polynomial time for a given topology.

We will demonstrate this constructive method, by using it to show that conditions (W1), (W2) and (W3b) are not sufficient to guarantee that a minimum Gilbert network is a tree — this is why we included all four conditions in the definition of the Gilbert arborescence problem. The claim can be seen by considering the following example of a Gilbert network problem with two sources and one sink in the Euclidean plane, where we will show that there exists a *split-route flow* (that is, some vertex has at least two outgoing edges and therefore the network topology contains a cycle) that has a lower cost than any arborescence.

For this example, assume there are two sources $a, b$ and a sink $q$ which are the vertices of a triangle $\triangle abq$ with edge lengths $|ab| = 1$ and $|aq| = |bq| = 10$, as illustrated

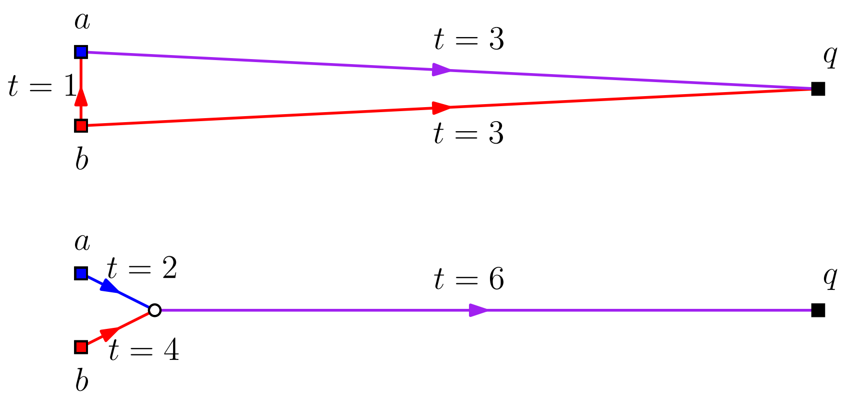


Figure 4.30: Two possible solutions for the example are shown. In each the flow from $a$ is indicated in blue, the flow from $b$ in red, and a combined flow in purple. The top diagram shows a split-route flow solution, while the bottom shows the only possible topology for a minimum Gilbert arborescence.

in Figure 4.30 (top). The magnitude of the flow from $a$ and $b$ is $2$ and $4$, respectively. The weight function is $w(f) = \lceil (3t+1)/2 \rceil$, i.e., $(3t+1)/2$ rounded up to the nearest integer. This function is positive, non-decreasing and subadditive, but not concave. For the example, only the following values need to be considered:

| $t$ | 1 | 2 | 3 | 4 | 6 |
|---|---|---|---|---|---|
| $w(f)$ | 2 | 4 | 5 | 7 | 10 |

Routing $1$ unit of the flow from $b$ to $q$ via $a$ (and the remainder of the flow directly to $q$) gives a Gilbert network (Figure 4.30 (top)) of total cost

$$w(1)|ab| + w(3)|aq| + w(3)|bq| = 102.$$

For a Gilbert arborescence the flows from $a$ and $b$ to $q$ are routed via some variable point $s$ as in Figure 4.30 (bottom). A minimum Gilbert arborescence can be calculated by using the weighted Melzak algorithm as described above. To construct the Steiner point $s$, we first construct the weighted equilateral point $e_{ab}$ outside $\triangle abq$ such that

$$|ae_{ab}| = \frac{w(4)}{w(6)}|ab| = 0.7 \quad \text{and} \quad |be_{ab}| = \frac{w(2)}{w(6)}|ab| = 0.4,$$

as shown in Figure 4.31. We then construct the circumcircle of $\triangle abe_{ab}$, which intersects the weighted Simpson line $e_{ab}q$ in the required point $s$. It follows from Theorem 4.37 that the resulting total cost is

$$\begin{aligned} & w(2)|as| + w(4)|bs| + w(6)|sq| \\ = \; & w(6)|e_{ab}q| = \sqrt{9982.5 + 7\sqrt{3890.25}} \\ = \; & 102.074\ldots \end{aligned}$$

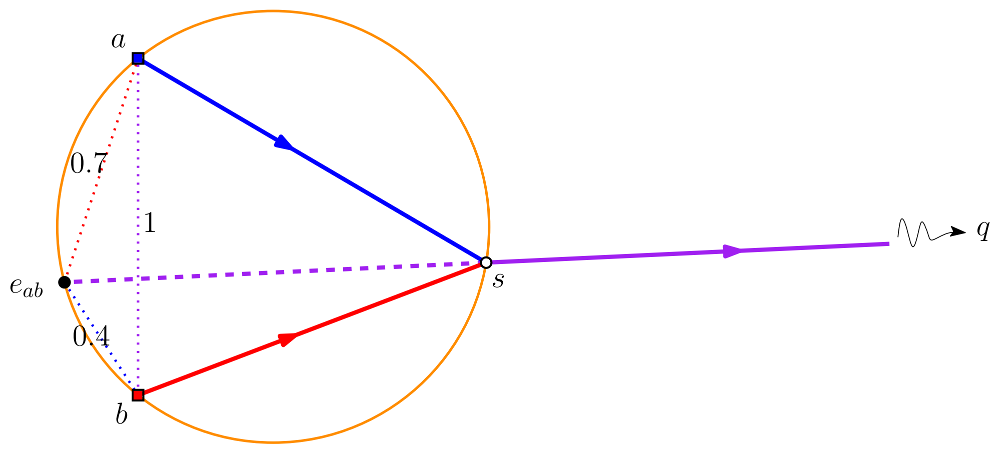


Figure 4.31: Construction of the Steiner point and weighted Simpson line for the minimum Gilbert arborescence for the example.

This shows that the split routing network constructed in Figure 4.30(top) is cheaper than the cheapest network with non-split routing, meaning that split routing can be necessary when the weight function is not concave. So, for minimum Gilbert arborescences we assume that the weight function $w$ satisfies conditions (W1), (W2), and (W3a) (and hence also (W3b)). If these conditions hold, it is known [179, 368] that in the case where there is a single sink there always exists a minimum Gilbert network that is a Gilbert arborescence.[55]

## Characterisation and degrees of Steiner points

From the above discussion, we now have a finite algorithm for constructing a Euclidean minimum Gilbert arborescence, using a Melzak-like strategy, as long as we know that all Steiner points in the minimum Gilbert arborescence have degree $3$. Hence, it is natural to ask the question: Under what conditions and for which weight functions does this degree $3$ property hold? Here we will summarise the main known results in this area, all of which have appeared in the paper of Volz et al. [379]. The proofs of these results require detailed analysis and a subtle understanding of dual vector spaces, and are not included here.

We first look at some theorems that show that in Euclidean space there is a very general class of weight functions for which the Steiner points of a minimum Gilbert arborescence are necessarily of degree $3$. Although our main interest is in networks in the plane, these results are in fact completely independent of the dimension.

This first theorem gives a general characterisation of the degree of Steiner points in a minimum Gilbert arborescence.

**Theorem 4.38** *A Steiner point of degree $m+1$ is possible in some minimum Gilbert arborescence in Euclidean space with a weight function $w(\cdot)$ that satisfies conditions* (W1)–(W3a)*, if and only if there exist vectors $\mathbf{v}_1, \dots, \mathbf{v}_m$ and flow demands $f_1, \dots, f_m > 0$ such that*

$$|\mathbf{v}_i| = w(f_i), \qquad i = 1, \dots, m \tag{4.6}$$

$$|\sum_{i=1}^{m} \mathbf{v}_i| = w(\sum_{i=1}^{m} f_i), \tag{4.7}$$

$$\forall I \subseteq \{1, \dots, m\} \textit{ with } 2 \leq |I| \leq m-2, \quad |\sum_{i \in I} \mathbf{v}_i| \leq w(\sum_{i \in I} f_i). \tag{4.8}$$

In this theorem the vectors $\mathbf{v}_1, \dots, \mathbf{v}_m$ correspond to the $m$ edges directed towards a given Steiner point, and the $f_i$'s are the associated flows. By Kirchhoff's law the flow on the outward edge of the Steiner point is $\sum_{i=1}^{m} f_i$. Hence, we can think of Equations (4.6) and (4.7) as constituting a flow-balancing condition at the Steiner point. Equation (4.8) guarantees that there is no advantage in any subset of the incoming flows combining at another point before they reach the Steiner point; in other words, there is no cost-reducing perturbation under which the Steiner point splits into two or more Steiner points.

Using Theorem 4.38 it is possible to define a quite general class of weight functions for which all Steiner points are necessarily of degree $3$.

**Theorem 4.39** *If the weight function $w(\cdot)$ satisfies conditions* (W1), (W2), (W3a) *and is differentiable with $w(0) > 0$ and such that $(w^2)'$ is a convex function, then all Steiner points in minimum Gilbert arborescences with weight function $w$ in Euclidean space have degree $3$.*

It easily follows that the degree $3$ condition holds for any positive linear weight function. Indeed, it is straightforward to show that the hypothesis of Theorem 4.39 is satisfied for the weight function $w(f) = d + hf^{\alpha}$ for any $d, h > 0$ and $\alpha \in [0, 1/2] \cup \{1\}$, but not when $\alpha \in (1/2, 1)$ (Exercise 4.16). For any $\alpha \in (1/2, 1)$ it has been shown that there exist weight functions of the above form for which a minimum Gilbert arborescence can have a degree $4$ Steiner point.

Note that in proving Theorem 4.39 one has to inductively apply Theorem 4.38 to all values of $m + 1 \geq 3$. Again the details of the proof appear in [379].

Finally, in a smooth Minkowski plane we have the following result.

**Theorem 4.40** *Let $w(\cdot)$ be a linear weight function $w(f) = d + hf$, $d > 0, h \geq 0$. In a smooth Minkowski plane, a Steiner point in a minimum Gilbert arborescence with cost function $w$ necessarily has degree $3$.*

### 4.4.2 Applications and extensions

There are many applied Steiner tree problems in which the presence of a flow on the network should ideally be incorporated into the optimisation model, either as part of the objective function or as constraints. The original application given by Gilbert [179] was to communications networks, where the weight associated with an edge of the network indicates the number of channels being installed along the corresponding route. One of the most significant current applications where the effects of flow need to be taken into account is in the physical design of microchips; here the geometry of the network affects the signal delay and skew of each net, which can strongly impact the performance of the chip. This has been discussed in Section 3.6.2 of Chapter 3. Another recent application where signal flow is potentially important is in the design of wireless sensor networks; some aspects of this will be examined later in this chapter in Section 4.5.1.

In the remainder of this section we briefly look at two flow-related Steiner tree problems that are closely associated with engineering applications: grade of service Steiner trees, and gradient-constrained Steiner trees with flows.

#### Grade of service Steiner trees

Suppose we are given a collection of cities (modelled as points on a plane) that we wish to interconnect by a minimum cost network of roads. Solving the Euclidean Steiner tree problem for the given set of cities gives a minimum total length to the road network, but does not account for the fact that not all roads are of equal cost; the cost, per kilometre, of building and maintaining a multi-lane highway is much greater than that for a road with a single lane in each direction. Each edge of the Steiner tree should be weighted to indicate the *grade of service* of the section of road associated with that edge; the weights in the tree should be proportional to the per-kilometre costs of the grade of road for each edge. The grade of service associated with each edge is determined by the volume of traffic using each section of road — this can be determined by the volume of traffic driving in and out of each city. In particular, for each city $t_k$ we can associate a *service request grade* $r(t_k) \in \mathbb{R}^+$ so that for the path in the tree between any two given cities $t_i$ and $t_j$ every edge of the path has a grade of service of at least $\min\{r(t_i), r(t_j)\}$. This problem can be formally stated as follows.[56]

EUCLIDEAN GRADE OF SERVICE STEINER TREE PROBLEM IN THE PLANE
**Given**: A set of points (terminals) $N = \{t_1, \ldots, t_n\}$ lying in the plane, and a service request grade $r(t_k) \in \mathbb{R}^+$ for each terminal $t_k \in N$.
**Find**: A geometric network $T = (V(T), E(T))$, satisfying the following properties: $N \subseteq V(T)$; each edge $e \in E(T)$ has an assigned grade of service $r(e)$ such that, for every pair of terminals $t_i$ and $t_j$, if $e$ lies on the path in $T$ between $t_i$ and $t_j$, then $r(e) \geq \min\{r(t_i), r(t_j)\}$; and $\sum_{e \in E(T)} r(e)|e|$ is minimised.

A network $T$ solving the grade of service Steiner tree problem is a tree, referred to as a *minimum cost grade of service Steiner tree*. If the topology of $T$ is known, then the grade of service for every edge of $T$ can be determined in linear time.

**Lemma 4.41** *[418] Given any Steiner topology $\mathcal{T}$ for $N$, we can compute the grades of service for all edges that appear in a minimum cost grade of service Steiner tree with topology $\mathcal{T}$ in $O(n)$ time.*

The idea behind this result is that deleting any edge $e$ of $\mathcal{T}$ creates two connected components, and hence partitions $N$ into two subsets. By taking the largest service request grade associated with each subset and then taking the minimum of these two values we obtain the required grade of service of $e$. Using this principle, it is reasonably straightforward to prove Lemma 4.41 (Exercise 4.17).

As discussed in Section 4.4.1, however, knowing the topology and edge weights of a minimum grade of service Steiner tree may not be enough to allow us to efficiently construct the tree (even if we add the constraint that every Steiner point has degree $3$) — whether the tree can be constructed in polynomial time is currently an open problem. Xue et al. [418] have developed a heuristic algorithm and a recursive approximation algorithm for this fixed topology problem, as well as a general branch-and-bound approach (see also Chapter 4 of [135]).

### Gradient-constrained networks with flows

An important application of the Gilbert arborescence problem in the plane, already mentioned in Section 4.1.3, is to the strategic design of underground mine access tunnels in a vertical plane. Here each of the source nodes represents a stope or stoping level and the sink represents the mine portal; the flow demands correspond to the total tonnages of ore transported along the access tunnels over the life of the mine, as specified in Equation (4.1). Note that the resulting weight on each edge of the network is a linear function of the development and haulage costs, and hence a linear function of the total tonnage.

The metric used in this model for measuring the length of each edge is the gradient-constrained metric, since there is a maximum gradient at which the haulage trucks can travel.

Unlike minimum Steiner trees, minimum Gilbert arborescences in a vertical plane under this metric do not appear to have a strong geometric structure or a canonical form. However, it has been shown in [378] that there are structural restrictions at Steiner points that have the potential to be exploited as part of an exact or heuristic algorithm. For a given gradient-constrained Gilbert arborescence with maximum gradient $m$, suppose we label each edge as follows: 'b' to indicate that the edge is bent; 'm' to indicate that the edge is straight with absolute gradient $m$; or 'f' (for flat) to indicate that the edge is straight with absolute gradient less than $m$. Then, in terms of these labels, we have the following key result.

**Theorem 4.42** *[378] Let $T$ be a minimum gradient-constrained Gilbert arborescence and let $s$ be a Steiner point in $T$. Then:*

1. *The Steiner point $s$ has at most one incident b-edge.*
2. *If the incident sink edge to $s$ is an m-edge, then $s$ has at most one incident f-edge.*
3. *The degree of $s$ is either 3 or 4.*

It is also shown in [378], using methods of subdifferential calculus and Minkowski sums, that if Steiner points are classified by the labels of their incident edges, then there is only a small number of possible label combinations that can occur in a minimum gradient-constrained Gilbert arborescence (that is, a smaller number of possibilities than those allowed by Theorem 4.42). See [71] for further details, where it is also shown that similar results can be derived for the problem in three dimensions.

## 4.5 Related topics

In this section we briefly describe a few other topics. These are either more recent or less well-developed variations on the Steiner tree problem, but ones that are natural generalisations of the standard problem from a mathematical point of view or are constrained versions that have interesting potential applications.

## 4.5.1 Power-$p$ Steiner trees

In most of the Steiner problems we have considered so far, the cost of an edge is directly proportional to its length (or magnitude under a given metric). However, for some applications, particularly those related to wireless communications, the cost of an edge can be more accurately modelled in terms of its length raised to a given power, $p$.

POWER-$p$ STEINER TREE PROBLEM IN THE PLANE
**Given**: A set of points $N = \{t_1, \ldots, t_n\}$ lying on a normed plane (with norm denoted by $\|\cdot\|$), and a real number $p$.
**Find**: A geometric network $T = (V(T), E(T))$, such that $N \subseteq V(T)$, and such that the cost $\sum_{e \in E(T)} \|e\|^p$ is minimised.

For applications relating to wireless communication networks, where the cost of an edge is proportional to the transmission power required to communicate between two adjacent vertices, $p$ is generally modelled as a real number between $2$ and $5$ (see, for example, [232]). In such applications $p$ is sometimes also known as the *path loss exponent*.

In a similar way to previous sections, we refer to the network $T$, which necessarily has a tree topology, as a *minimum power-$p$ Steiner tree*, and to the elements of $V(T)\backslash N$ as Steiner points. We have previously discussed a version of this problem as an example of a generalised $k$-Steiner tree problem (in Section 4.3.1). Here again, for $p > 1$, a minimum Steiner tree will only exist if there is some sort of bound on the number of Steiner points, since otherwise the cost of any network can be reduced by adding one or more degree $2$ Steiner points to any edge (Exercise 4.18). However, rather than simply imposing a fixed bound on the number of Steiner points, as in Section 4.3.1, one can adopt other methods for constraining the number of Steiner points. Two popular methods are to impose a lower bound on the degree of Steiner points, by stipulating, for example, that the degree of each Steiner point is at least $3$, or alternatively to impose a fixed cost on each Steiner point which is included as part of the cost function of the Steiner tree.[57]

For most values of $p$, the power-$p$ Steiner tree problem appears to be difficult to solve exactly. For example, Ganley and Salowe [167] strongly conjecture that the problem is not finitely solvable for integer values of $p \geq 5$. However, the $p = 2$ case, which is sometimes referred to as the *quadratic Steiner tree problem,* does admit exact solutions, which can be found either algebraically or geometrically.

The Euclidean quadratic Steiner tree problem was studied by Ganley and Salowe, in [161] and [167]. They showed that in a minimum quadratic Steiner tree a Steiner point of degree $3$ is located at the arithmetic mean of the three neighbouring vertices. In other words, if the neighbouring vertices have coordinates $(a_x, a_y), (b_x, b_y), (c_x, c_y)$, respectively, then the Steiner point has coordinates $(a_x + b_x + c_x, a_y + b_y + c_y)/3$. This means that given a full Steiner topology (where each Steiner point has degree 3) for a quadratic Steiner tree, the coordinates for all Steiner points can be expressed in terms of a system of linear equations. Ganley and Salowe show that these equations are well conditioned, and hence the system can be solved in linear time (using, for example, Gaussian elimination) in terms of the coordinates of the terminals. Unfortunately, the large number of possible topologies makes this an impractical approach for solving the quadratic Steiner tree problem for more than a small number of terminals.

**Flow-dependent quadratic Steiner trees**

A natural way of generalising the Euclidean quadratic Steiner tree problem, particularly in terms of wireless sensor network applications, is to combine it with the Gilbert arborescence problem, described in Section 4.4.1. If the cost of the tree aims to model power consumption in a wireless sensor network, then this cost should take into account the amount of data flowing from each sensor to the base station. As with the Gilbert arborescence problem, we thus think of the set of terminals $N$ as consisting of $n$ sources (each with a given flow supply) and a single sink. For any specific tree $T$ spanning the terminals and a set of Steiner points we can uniquely associate with each edge $e_i$ a flow $f_i$ satisfying the given flow supply at each source and conservation of flow at each Steiner point. Here we take the cost of each edge to be the flow on that edge times the square of its length, and hence the flow-dependent cost of $T$ is

$$|T|_f := \sum_{e_i \in E(T)} f_i |e_i|^2.$$

The *flow-dependent quadratic Steiner tree problem* is the problem of finding a tree $T$ spanning $N$ with minimum flow-dependent cost; such a tree is referred to as a *minimum flow-dependent quadratic Steiner tree*.

The flow-dependent quadratic Steiner tree problem has been studied in some detail in [57]; here we survey some of the key results. Given a Steiner point $s$ in a minimum flow-dependent quadratic Steiner tree $T$, we can assign a weight to each neighbouring vertex $v_i$ of $s$ in $T$, namely the flow on the edge between $v_i$ and $s$. Note that determining the weight of each vertex $v_i$ depends only on knowing the topology of $T$ (but not the location of any of the Steiner points). Using this definition of weight we have the following theorem.

**Theorem 4.43** *[57] In a minimum flow-dependent quadratic Steiner tree each Steiner point lies at the centre of mass of its neighbouring vertices.*

This theorem gives a simple geometric method for finding the location of a Steiner point in terms of the positions of its neighbouring vertices, using properties of mass point

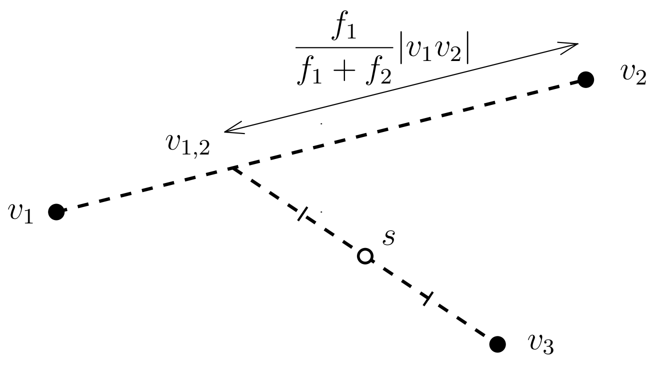


Figure 4.32: Merging weighted vertices to construct a Steiner point. Here $v_1, v_2, v_3$ have weights $f_1, f_2, f_1 + f_2$, respectively.

geometry (see, for example, [125]). The centre of mass of a given set of points can be constructed geometrically by recursively merging masses and subdividing line segments into appropriate ratios. For example, consider the problem of constructing a Steiner point $s$ of degree $3$ with neighbouring vertices $v_1, v_2, v_3$, where the flow is directed towards $v_3$, and where the neighbouring vertices have weights $f_1, f_2, f_1 + f_2$, respectively; this is illustrated in Figure 4.32. We first merge $v_1$ and $v_2$ into the point $v_{1,2}$ where $v_{1,2}$ is the point on the line segment $v_1v_2$ such that $|v_1v_{1,2}|/|v_2v_{1,2}| = f_2/f_1$. This merged point $v_{1,2}$ is assigned the weight $f_1 + f_2$. Merging $v_{1,2}$ and $v_3$ yields the point $s$ at the midpoint of $v_{1,2}v_3$, since $v_{1,2}$ and $v_3$ have equal weights.

In the above merging process, the point $v_{1,2}$ essentially plays the same role as a pseudo-terminal in the merging phase of the Melzak-Hwang algorithm for the Euclidean Steiner problem (Section 1.2.1). Using the same recursive approach as in the Melzak-Hwang algorithm it is possible to describe a geometric linear-time algorithm for constructing a minimum flow-dependent quadratic Steiner tree with a given full topology, where each Steiner point is assumed to have degree $3$. Some care, however, needs to be taken in assigning weights to each of the new pseudo-terminals so that the recursive merging and reconstruction processes correctly construct the required Steiner points. Details on how to do this are given in [57]. This approach suggests that it should be possible to design a GeoSteiner-type algorithm for solving the flow-dependent quadratic Steiner tree problem (where Steiner points have maximum degree $3$).

### 4.5.2 Node-weighted Steiner trees

One of the methods mentioned in the previous section for effectively bounding the number of Steiner points (for power-$p$ Steiner trees) is to impose a cost on each Steiner point. This constraint can also be applied to other forms of the Steiner problem, such as the Steiner tree problem in a Euclidean or other normed plane. In terms of applications, the constraint is a natural one: for example, in wireless sensor networks the Steiner points represent the relays of the network, which carry significant construction, installation and maintenance costs; while in chip design the Steiner points correspond to the vias between layers which contribute to the overall length of the nets between modules in the chip.

This problem is known as the *node-weighted Steiner tree problem*. In describing the problem, we can assume that the only node costs that need to be considered are those associated with Steiner points; any costs placed on the terminals have no effect on the optimisation, since the terminals are fixed. This problem has mainly been studied for Steiner trees in graphs [346], and is closely related to the prize-collecting Steiner tree problem. The geometric version is defined as follows:

NODE-WEIGHTED STEINER TREE PROBLEM IN THE PLANE
**Given**: A set of points $N = \{t_1, \ldots, t_n\}$ lying on a normed plane (with norm denoted by $\|\cdot\|$), and a positive real number $p$.
**Find**: A geometric network $T = (V(T), E(T))$, such that $N \subseteq V(T)$, and such that the cost $p|V(T) \setminus N| + \sum_{e \in E(T)} \|e\|$ is minimised.

As usual, such a tree $T$ is known as a *minimum node-weighted Steiner tree* and the elements of $V(T) \setminus N$ are referred to as *Steiner points*.

For the remainder of this section we consider the Euclidean node-weighted Steiner tree problem. We first observe that if a minimum node-weighted Steiner tree has a Steiner topology (that is, a topology where each Steiner point has degree $3$), then $T$ is the same as the Euclidean Steiner tree for that topology (without node weights). More generally, once the topology of $T$ is known the node weight $p$ plays no further role in determining the locations of the Steiner points. However, unlike the standard Euclidean Steiner tree problem, it is possible for Steiner points of $T$ to have degree greater than $3$. An example is illustrated in Figure 4.33, where the terminals are the vertices of a $10 \times 10$ square, and $p = 1$. Here the minimum node-weighted Steiner tree contains a single degree 4 Steiner point; see Exercise 4.19.

The following theorem, proved by Rubinstein et al. [331] using variational techniques, shows that Steiner points of degree greater than $4$ never occur.

**Theorem 4.44** *[331] No Steiner point of degree* $5$ *or more can occur in a minimum node-weighted Steiner tree in the Euclidean plane.*

This result was also proved independently and extended to higher dimensions by Colthurst et al. [117]. It has also been shown that for any given instance there exists a minimum node-weighted Steiner tree in which each terminal has degree at most $5$ [416].

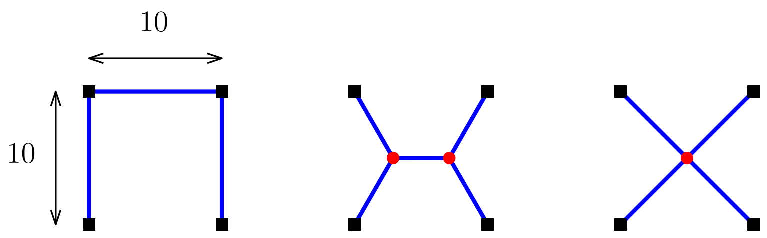


Figure 4.33: Examples of locally minimal interconnection trees for the vertices of a $10 \times 10$ square. If the Steiner points (indicated in red) each have a cost of 1 unit, then the cost of each tree (from left to right) is: $30$; $2 + 10(1 + \sqrt{3}) \approx 29.3205$; and $1 + 20\sqrt{2} \approx 29.2843$. Here the example on the right with a degree 4 Steiner point is minimum.

Theorem 4.44 is a very useful result, since locally minimal points of degree $5$ are not in general constructible using elementary geometric techniques; see for example [114]. Degree $3$ and degree $4$ Steiner points, on the other hand, can be easily constructed: for degree $3$ Steiner points, the three meeting angles are each $2\pi/3$ and the Steiner point can be constructed in terms of the locations of its neighbouring vertices, as detailed in Section 1.1; for degree $4$ Steiner points, the four incident edges form two collinear pairs, making the Steiner point trivial to construct in terms of its neighbouring vertices. A straightforward algorithm for constructing a node-weighted Steiner tree for a given topology (where each Steiner point has degree $3$ or $4$), based on the Melzak-Hwang algorithm, is given in [377] — see also Exercise 4.13. Unfortunately, this algorithm runs in exponential time for a given topology; the question of whether or not there exists a polynomial-time algorithm remains open.

Despite these challenges, a reasonably efficient algorithm for solving the node-weighted Steiner tree problem has been proposed and implemented by Xue [416]. Xue gives a method for enumerating the possible topologies, and proves a recursive bounding condition on subtrees; together these form the building blocks for an effective branch-and-bound algorithm, similar in strategy to Smith's algorithm [354]. In practice the algorithm is able to exactly solve problems with up to ten terminals.

## Minimum Steiner point trees

We conclude this section by describing a useful variation on the node-weighted Steiner tree problem. Suppose we consider an instance of the node-weighted Steiner tree problem in which the node weight $p$ is very large. In this case, the main optimisation objective is to minimise the number of Steiner points, and in the limit, as $p \to \infty$, the problem becomes the minimum spanning tree problem. If we then add a further constraint, namely a given upper bound on the length of any edge in the network, then the resulting problem is known as the minimum Steiner point tree problem. The formal definition is as follows.

MINIMUM STEINER POINT TREE PROBLEM IN THE PLANE
**Given**: A set of points $N = \{t_1, \ldots, t_n\}$ lying on a normed plane (with norm denoted by $\|\cdot\|$), and a positive real number $R$.
**Find**: A geometric network $T = (V(T), E(T))$, such that $N \subseteq V(T)$ and $\|e\| \leq R$ for each $e \in E(T)$, and such that $|V(T)|$ is minimised.

We refer to $T$ as a *minimum Steiner point tree* and to the elements of $V(T) \setminus N$ as Steiner points. Note that here, unlike many other variations of the Steiner tree problem, Steiner points may have degree $2$.

The edge-length bound is a very natural one in many applications. In wireless sensor networks it may correspond to the maximum distance a signal can be transmitted from the sensors and relays [93]; in optical networks it may represent a bound on the maximum distance between optical amplifiers [255, 322]; while in networks in microchips the bound may correspond to a maximum permitted distance between buffers [272].

The minimum Steiner point tree problem was first proposed by Sarrafzadeh and Wong [340], who showed that the problem is NP-hard in both the Euclidean and rectilinear metrics. Note that the problem is essentially the dual to the bottleneck $k$-Steiner tree problem (Section 4.3.3): here instead of bounding the number of Steiner points and minimising the longest edge length we place a bound on the longest edge length and minimise the number of Steiner points. This suggests a finite algorithm for solving the minimum Steiner point tree problem involving solving the bottleneck $k$-Steiner tree problem for different values of $k$ until the smallest $k$ for which the bottleneck length is no greater than $R$ is found.

Most of the literature on this problem has focussed on finding polynomial time approximation algorithms. These approximation algorithms are mainly based on *Steinerising* a minimum spanning tree, in other words, generating a minimum spanning tree and replacing any edge longer than $R$ by a path with the minimum number of degree $2$ Steiner points such that each edge in the path has length at most $R$ [263]. Măndoiu and Zelikovsky [275] have shown that this gives an approximation ratio of $4$ in the Euclidean plane and $3$ in the rectilinear plane. Chen et al. [84] have developed an improved polynomial-time approximation scheme for the Euclidean metric with a performance ratio of $3$.

An alternative approach, investigated by Brazil et al. [53], is to Steinerise a minimum Steiner tree rather than a minimum spanning tree. For any norm, this results in a solution that is larger than the optimal by at most $2n - 4$, where $n$ is the number of terminals. Of course, this is not a polynomial-time algorithm, since the Steiner tree problem itself is NP-hard, but for many norms this approach will work well in practice, using the GeoSteiner-type algorithms described in this book.

### 4.5.3 Rotationally optimal Steiner trees

The final problem we consider in this chapter is a variation on the $\lambda$-geometry Steiner tree problem (see Chapter 2), which includes the rectilinear Steiner tree problem. In these problems the minimum Steiner tree is composed of line segments, each of which uses one of a set of $\lambda$ evenly spaced orientations. For the $\lambda$-geometry Steiner tree problem this set of orientations is given, but suppose, instead, we are permitted to choose any set of $\lambda$ evenly spaced orientations; in other words, we can rotate all legal orientations simultaneously to find the set that gives the minimum Steiner tree with shortest length. This results in the following problem.

ROTATIONALLY OPTIMAL UNIFORM ORIENTATION STEINER TREE PROBLEM IN THE PLANE
**Given**: A set of points $N = \{t_1, \ldots, t_n\}$ lying in the plane, and a positive integer $\lambda \geq 2$.
**Find**: A geometric network $T = (V(T), E(T))$ and a uniform orientation metric $\|\cdot\|_\lambda$ with $\lambda$ legal orientations, such that $N \subseteq V(T)$ and such that $\sum_{e \in E(T)} \|e\|_\lambda$ is minimised.

As discussed in Section 2.7, the uniform orientation Steiner tree problem has important applications in chip design and printed circuit board design. The rotationally optimal version of this problem has the potential to further optimise such networks and to provide an accurate lower bound on the minimum length possible. However, given that we are considering a continuous set of rotations of the coordinate system defining the legal orientations, it is not immediately obvious how to solve this problem, or indeed whether a finite algorithm exists.

The rectilinear problem (where $\lambda = 2$) has been studied by Nielsen et al. [298], and the more general uniform orientation problem by Brazil et al. [51]; we briefly survey the results of those papers here. Let $\omega = \pi/\lambda$. Note first, by symmetry, that the length of a $\lambda$-geometry network remains unchanged if we rotate the coordinate system by $\omega$ or a multiple of $\omega$. This means that we only need to consider rotations $\alpha$ in the interval $[0, \omega)$. Let $u$ and $v$ be a fixed pair of points in the plane such that $uv$ is in a legal orientation when $\alpha = 0$. Let $\|uv\|_\alpha$ be the length of $uv$ under the $\lambda$-geometry metric for a given $\alpha$. Then it is straightforward to see that

$$\|uv\|_\alpha = |uv| \frac{\sin \alpha + \sin(\omega - \alpha)}{\sin \omega}$$

where $|uv|$ is the Euclidean length of $uv$. (In particular, when $\lambda = 2$ and hence $\omega = \pi/2$ we have $\|uv\|_\alpha = |uv|(\sin \alpha + \cos \alpha)$, as expected.)

It follows that the function $f_{uv}(\alpha) = \|uv\|_\alpha$ is periodically strictly concave (with period $\omega$) and that the minimum is attained when $uv$ has a legal orientation. This immediately gives us a finite method for solving the *rotationally optimal uniform orientation minimum spanning tree problem* (where no Steiner points are permitted). Since a sum of strictly concave functions is strictly concave, it is easy to see that in a rotationally optimal uniform orientation minimum spanning tree at least one of the edges of the tree must be straight (that is, have a legal orientation). Hence, an algorithm for solving the problem is as follows: for each pair of terminals $t_i, t_j$ in $N$ choose $\alpha$ so that $t_i t_j$ has a legal orientation and then solve the uniform orientation minimum spanning tree problem for this fixed $\alpha$ — the minimum tree over all pairs is the solution. This algorithm runs in time $O(n^3 \log n)$.

In [51] it is shown that a similar result holds for the rotationally optimal uniform orientation Steiner tree problem. Recall, from Chapter 2, that for the uniform orientation Steiner tree problem there exists a solution in which each full component $F$ contains at most one bent edge (consisting of two *half-edges*, which are the line segments between the corner point and a vertex) — all other edges are straight edges. It can be shown that as the coordinate system is continuously rotated by $\alpha$ (over a suitable interval) the length of each straight edge and half-edge of $F$ defines a strictly concave function of $\alpha$. This implies the following theorem.

**Theorem 4.45** *[51] A rotationally optimal uniform orientation minimum Steiner tree is a union of full Steiner trees, at least one of which contains no bent edges.*

This theorem leads to a finite algorithm for constructing the optimal tree. For every subset $N_i$ of the terminals, every Steiner topology $\mathcal{T}$ on $N_i$, and every legal angle distribution around Steiner points in $\mathcal{T}$ for the given $\lambda$, a full Steiner tree in $\lambda$-geometry without bent edges can be constructed in a bottom-up manner (if it exists). If such a tree exists, the orientations used by the edges in the tree give a feasible rotation angle $\alpha$, for which the uniform orientation Steiner tree problem for that fixed value of $\alpha$ can be solved in finite time.

For general $\lambda$ the algorithm is rather impractical, as a super-exponential number of rotation angles must be tried. However, for the rectilinear case ($\lambda = 2$) only $O(n^2)$ rotation angles need to be considered: those given by the lines through each pair of terminals. This follows from the fact that for the rectilinear plane each full component of a minimum Steiner tree can be assumed to be in Hwang form. When there is no bent edge in such a full component, the backbone becomes a line segment between two terminals.

Shang et al. [347] have also studied the Steiner ratio for the rotationally optimal uniform orientation Steiner tree problem. Recall, from Section 2.5.1, that for a given $\lambda$, if $T_\lambda(N)$ and $\overline{T}_\lambda(N)$ denote a minimum Steiner tree and a minimum spanning tree, respectively, for $N$ in $\lambda$-geometry, then the Steiner ratio is defined as follows:

$$\rho_\lambda = \inf_N \frac{\|T_\lambda(N)\|}{\|\overline{T}_\lambda(N)\|}.$$

The *rotationally optimal Steiner ratio* $\hat{\rho}_\lambda$ is defined in the same way except that $T_\lambda(N)$ and $\overline{T}_\lambda(N)$ are rotationally optimal minimum spanning trees and minimum Steiner trees respectively (each with potentially different rotation angles for the coordinate system). For the case where $|N| = 3$, Shang et al. [347] show that

$$\hat{\rho}_2 = \frac{3+\sqrt{3}}{6} \text{ and } \hat{\rho}_4 = \frac{\sqrt{6}-\sqrt{3}+1}{2},$$

and they conjecture that these ratios hold for any cardinality of $N$.

## Exercises

**4.1.** Prove Theorem 4.4. [Hint: See the paragraph immediately following the statement of Theorem 4.4.]

**4.2.** Verify Theorem 4.5 for the cases where the number of terminals of $T$ is two, three or four.

**4.3.** Construct an example consisting of two points and a single obstacle, with visibility graph $G$, satisfying the following properties: there is a unique minimum length obstacle-avoiding path $P_1$ in $G$ between the two points under the $L_1$ norm; there is a unique minimum length obstacle-avoiding path $P_2$ in $G$ between the two points under the $L_2$ norm; and $P_1$ and $P_2$ are distinct.

**4.4.** Prove Theorem 4.10 using any of the results proved or quoted in Section 1.6.

**4.5.** Prove Corollary 4.14.

**4.6.** Prove Lemma 4.16. [Hint: Show that there exists an embedding of $v_iv_{i+2}$ in the Euclidean plane (using legal orientations) that passes through $v_{i+1}$.]

**4.7.** Show that any convex obstacle in the rectilinear plane with five or more vertices contains at least one transparent vertex, and hence deduce Theorem 4.19.

**4.8.** Show that the cost functions $\alpha_p$, for $p > 0$, and $\alpha_\infty$ are $\ell_1$-optimisable.

**4.9.** Prove Lemma 4.21. [Hint: Given any point $y_0$ on $\mathrm{bd}(B)$ construct a translation of $\mathrm{bd}(B)$ centred around $y_0$. The point $y_0$, along with the two intersection points of the two boundaries closest to $y_0$, will constitute three of the six points. The remaining three points can be constructed using the central symmetry of $B$.]

**4.10.** Assume that the terminals $a$, $b$ and $c$ in Figures 4.22 and 4.23 have coordinates $(5,7)$, $(0,2)$ and $(4,0)$, respectively, and assume that the Euclidean hexagonal direction set includes the horizontal directions. Using a vector-based graphics program, or otherwise, construct the six $i$th ODC partitions for these three terminals and the resulting OODC partition.

**4.11.** Prove the claim in the proof of Theorem 4.28. [Hint: Let $t_i \in A_b \backslash \{t_r\}$ and consider separately the two cases where $t_{b+1}$ does and does not lie on $P_T(t_i, t_r)$. For the second case, let $y$ be the first common point of the paths $P_T(t_i, t_r)$ and $P_T(t_{b+1}, t_r)$, and make use of the observation that $\|\text{BSD}_T(t_i, y)\| \geq \|\text{BSD}_T(y, t_r)\|$.]

**4.12.** Prove that each network $T^i$ generated in the F-SMT algorithm (Algorithm 4.2) is a tree. The proof should be by induction on the number of connected components in the input viable forest $F$, noting that the base case, $i = 1$, follows from Theorem 4.28.

**4.13.** Find a finite algorithm for solving the fixed topology Euclidean Steiner tree problem in the plane, where Steiner points can have degree $3$ or $4$. [Hint: First show that for a degree $4$ Steiner point the four incident edges form two collinear pairs; in other words, the edges in the neighbourhood of the Steiner point look like two crossing lines. Hence, show that each degree $4$ Steiner point corresponds to one of three possible decompositions of the problem into two smaller problems.]

**4.14.** Prove Lemma 4.31 by showing that it is a corollary to Theorem 4.27.

**4.15.** Let $s$ be a degree $3$ Steiner point of a Euclidean minimum Gilbert network $T$, with meeting angles $\alpha$, $\beta$ and $\gamma$. If $w_1$, $w_2$ and $w_3$ are the weights of the three edges incident with $s$, show that the meeting angles satisfy equations (4.3), (4.4) and (4.5), and that the angles of the weight triangle of $s$ are $\pi - \alpha$, $\pi - \beta$ and $\pi - \gamma$.

**4.16.** Show that the hypothesis of Theorem 4.39 is satisfied for the weight function $w(t) = d + ht^\alpha$ for any $d, h > 0$ and $\alpha \in [0, 1/2] \cup \{1\}$, but not when $\alpha \in (1/2, 1)$.

**4.17.** Prove Lemma 4.41.

**4.18.** For $p > 1$, compute the cost $c$ of a straight-line path between two fixed points $p$ and $q$ in a Euclidean power-$p$ Steiner tree containing $k$ equally spaced Steiner points between $p$ and $q$. Show that $c \to 0$ as $k \to \infty$.

**4.19.** Let $N$ be the four vertices of a $10 \times 10$ square. Show carefully that the minimum node-weighted Steiner tree, with $p = 1$ for $N$, is the rightmost tree shown in Figure 4.33. Furthermore, for each of the trees illustrated in Figure 4.33 determine the range of values of $p$ for which it is a minimum node-weighted Steiner tree for $N$.

# Notes

[42]This background on mining, which focusses on hard rock mines containing metallic deposits, is drawn primarily from [8] and [48]. For a more comprehensive introduction to the infrastructure of an underground mine see [188].

[43]Some examples of the many heuristic approaches developed for the rectilinear obstacle-avoiding Steiner tree problem include the algorithms of Lin et al. [260], Long et al. [270], Li and Young [256], Liu et al. [265] and Ajwani et al. [5]. The last of these makes use of the FLUTE algorithm, described in Section 3.4. Also of note is a graph-based approximation algorithm for the octilinear obstacle-avoiding Steiner tree problem proposed by Müller-Hannemann and Schulze [291].

[44]For the equivalent result to Lemma 4.8 in the Euclidean plane, see for example [348].

[45]One of the most important early references on the Euclidean problem with polygonal obstacles is the paper of Provan [320] on approximation schemes for this problem. Other early papers relating to the Euclidean problem include [353] and [393]. The key papers on algorithmic approaches to solving the exact problem are those of Winter [405] and Zachariasen and Winter [435].

[46]Although numerous heuristics for the rectilinear minimum obstacle-avoiding Steiner tree problem have been developed, the literature on exact solutions is rather sparse. The first substantial results appear to be those of Ganley and Cohoon [164], who developed algorithms for solving instances with up to four terminals, using the so-called escape graph. More recently, Huang et al. [206, 210] presented a GeoSteiner-type algorithm able, in theory, to exactly solve instances of arbitrary size, using an approach similar to the one developed in this section. These methods were further refined and discussed by Juhl [227]. We note, however, that all of these previous results assume that the polygonal obstacles have boundary edges using only legal orientations, whereas we make no restrictions on the orientations of the boundary edges of obstacles in this section.

[47]A slightly weaker version of this result has appeared in the literature, but only for the rectilinear Steiner tree problem, with rectilinear obstacles; see [207], [208], [210] and [227].

[48]This application was first suggested by David Lee in his unpublished paper, 'Some industrial case studies of Steiner trees' presented at the NATO Advanced Research Workshop, 'Topological Network Design: Analysis and Synthesis', in Copenhagen, 1989.

[49]The bottleneck Steiner tree problem was first proposed by Chiang et al. [96] in the context of Steiner trees in graphs, and was originally known as the *Steiner min-max tree problem*. There it was shown that this problem has a simple polynomial-time algorithm, in terms of the number of vertices and edges of the entire graph. However, in terms of the cardinality of the terminal set, Berman and Zelikovsky [28] have shown that the problem does not admit any polynomial-time approximation algorithm with performance ratio less than 2 unless $P = NP$. The Euclidean version of the geometric bottleneck $k$-Steiner tree problem in the plane, as defined in this chapter, was first studied by Du et al. in [140] and [386]. A survey of the results in this area up to 2008, with a focus on approximation algorithms, can be found in Chapter 6 of [135].

[50]The algorithm developed in Section 4.3 for the general $k$-Steiner tree problem can in theory apply to a wider range of norms than the PE norms. The algorithm for these other norms will, however, be difficult to implement in practice. The precise required properties of the widest class of norms for which the algorithm can apply are given in [52].

[51]Note that [21] gives a slightly weaker version of condition 3, the bound on the degree of Steiner points; in particular, for $\ell_1$ and $\ell_\infty$ an upper bound of 7 rather than $5$ is given. The bounds in [21] are based on the Hadwiger numbers of the unit balls, that is, the largest number of non-overlapping translates of the unit ball

that can be brought into contact with it. However, for $\ell_1$ and $\ell_\infty$ (each of which have polygonal unit balls) a tighter bound can be found by using Lemma 4.22 together with the arguments of Monma and Suri in [288].

[52]This paper actually predates Gilbert's seminal paper with Henry Pollak [180] on the Euclidean Steiner tree problem.

[53]The problem of constructing a Steiner point in the weighted case (i.e., the weighted Fermat-Torricelli problem) was first solved, for the Euclidean plane, by Weber [392] in 1909. More details on the history and mathematics behind the general weighted Fermat-Torricelli problem (for Steiner points of degree $\geq 3$) can be found in the surveys of Wesolowsky [398] and Kupitz and Martini [242]. A recent treatment of the Euclidean weighted Fermat-Torricelli problem for Steiner points of degree $3$ can be found in a paper of Jalal and Krarup [224], where the problem is referred to as FERPOS.

[54]Ptolemy's theorem is named after the Greek astronomer and mathematician Claudius Ptolemaeus, who stated and used the result in about AD 150. Jalal and Krarup [224] prove Theorem 4.37 without the use of Ptolemy's theorem, and are then able to obtain Ptolemy's theorem as a simple corollary.

[55]Note that in [124], condition (W3b), the subadditivity condition, was incorrectly interpreted as concavity of the cost function.

[56]The formulation and solution of a hierarchical network design problem in which there are at least two grades of service dates back to a 1986 paper of Current et al. [126]. The first study of the grade of service Steiner tree problem, where the grades of service on the edges are determined by weights on the terminals, was a paper by Colbourn and Xue in 2000 [116], although this was for the Steiner tree problem in graphs. The main study of the geometric version of this problem is the paper of Xue et al. [418]. Note that our formulation of the Euclidean grade of service Steiner tree problem in the plane slightly simplifies that given in [418].

[57]The former of these two methods of bounding the number of Steiner points is the most popular in the literature, and is employed in [358], [161], [167] and [28] amongst others. The second method is particularly relevant to applications surrounding the modelling of wireless sensor network deployment; see for example [414] and [415]. Both methods are also discussed in [57].

# Chapter 5

# Steiner Trees in Graphs and Hypergraphs

In this chapter we consider two combinatorial versions of the Steiner tree problem: the Steiner tree problem in *graphs* and the Steiner tree problem in *hypergraphs*. Also, we consider the minimum spanning tree problem in hypergraphs. Although this book focuses on geometric interconnection problems in the plane, these combinatorial problems are included for several reasons. Firstly, the Steiner tree problem in graphs is probably the best studied of all the many variants of the Steiner tree problem. Secondly, the fixed orientation Steiner tree problem in the plane (and specifically the rectilinear Steiner tree problem in the plane) can be reduced to the Steiner tree problem in graphs (see Chapters 2 and 3). Thirdly, the full Steiner tree concatenation phase of GeoSteiner, the most efficient exact algorithm for computing minimum Steiner trees in the plane, can be reduced to either the Steiner tree problem in graphs or the minimum spanning tree problem in hypergraphs (see Section 1.4.4 in Chapter 1).

The purpose of this chapter is mainly to give a brief introduction, and to describe some of the most successful *exact* algorithms for these problems. The text is therefore not comprehensive, and most mathematical proofs are not included here; either they are left as exercises, or references to the original papers containing the proofs are given.

In contrast to Steiner tree problems in the plane, the set of Steiner points is a *given finite* set for Steiner tree problems in graphs and hypergraphs. This turns the problem into a pure combinatorial problem, where the task is to select an optimal subset of edges and/or vertices. Often methods other than those introduced so far in this book are required. The book by Korte and Vygen [239] gives an introduction to these methods. The books by Hwang, Richards and Winter [213], Prömel and Steger [319], and Du and Hu [135] give in-depth introductions to the Steiner tree problem in graphs; the paper by Polzin and Vahdati Daneshmand [314] covers recent algorithmic developments for the problem.

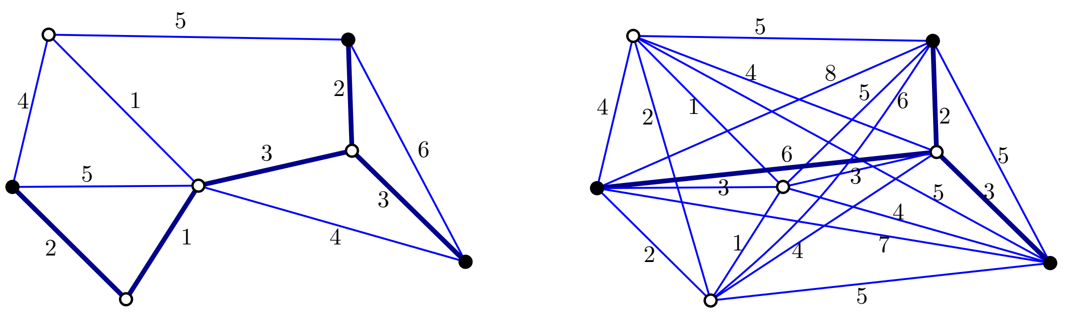


Figure 5.1: An example of the Steiner tree problem in graphs. The terminals are indicated by the solid black vertices, and the minimum Steiner tree is indicated by the bold edges. The diagram on the right shows the corresponding distance graph, with a minimum Steiner tree again indicated by bold edges.

## 5.1 Steiner trees in graphs

The Steiner tree problem in graphs is a well-studied combinatorial version of the geometric Steiner tree problem. The origins of the problem date back to the early 1970s.[58]

STEINER TREE PROBLEM IN GRAPHS
**Given**: An undirected, connected and edge-weighted graph $G = (V, E)$, where $c(e) > 0$ denotes the weight of edge $e \in E$, and a set of terminals $N_G \subseteq V$.
**Find**: A connected subgraph $T = (V(T), E(T))$ in $G$, such that $N_G \subseteq V(T)$, and such that $\sum_{e \in E(T)} c(e)$ is minimised.

Since the edge weights are positive, it is easy to see that the subgraph $T$ must in fact be a *tree*. Even if the edge weights are allowed to be zero or negative, the problem can be reduced to the problem with positive edge weights (see Exercise 5.1). A tree $T = (V(T), E(T))$ that represents a solution to the Steiner tree problem in graphs is called a *minimum Steiner tree*. An example of such a minimum Steiner tree is given in Figure 5.1.

The Steiner tree problem in graphs is NP-hard [233], and remains so for a number of special cases, including planar graphs [170] and complete graphs with edge weights 1 and 2 [30]. The problem has no polynomial-time approximation scheme unless P = NP [30].

The vertices $S(T) = V(T) \setminus N_G$ are denoted the *Steiner vertices* (or *Steiner points*) of $T$. Note that Steiner vertices in $T$ may have degree 2, that is, may be internal vertices of some path in $T$. Consider a path $P_T(u, v)$ in $T$ interconnecting two vertices $u, v \in V(T)$;

assume that all internal vertices (if any) on $P_T(u,v)$ are Steiner points of degree 2. Since $T$ has minimum weight, the path $P_T(u,v)$ must be a *shortest path* in $G$.

**Definition [Distance graph]**: The *distance graph* $D = (V(D), E(D))$ for an edge-weighted graph $G$ is a complete edge-weighted graph with vertex set $V(D) = V$. The weight of an edge $(u,v) \in E(D)$ is equal to the weight $c_P(u,v)$ of a shortest path between $u$ and $v$ in $G$.

An example of a distance graph is illustrated in Figure 5.1 (right). Note that edge weights in the distance graph have the properties of a metric: they are non-negative, symmetric, and they fulfil the triangle inequality. Based on these observations and definitions, we have the following lemma (compare to Theorem 1.2 in Chapter 1):

**Theorem 5.1** ***[Basic properties of a minimum Steiner tree in a graph]*** *Let $G = (V,E)$ be an undirected, connected and edge-weighted graph, and let $D$ be the distance graph for $G$. Let $T = (V(T), E(T))$ be a minimum Steiner tree for terminal set $N_G \subseteq V$ in $G$, where $n = |N_G|$. Let $\overline{S}(T) \subseteq S(T)$ be the set of Steiner vertices of degree 3 or more in $T$. Then $T$ satisfies the following conditions:*

*(1)* $|\overline{S}(T)| \leq n - 2$.

*(2) $T$ decomposes into at most $2n-3$ shortest paths (or* geodesics *in $G$) with endpoints in $N_G \cup \overline{S}(T)$.*

*(3) $T$ can be mapped into an equal-length tree in $D$ (and vice versa, any minimum Steiner tree in $D$ can be mapped into an equal-length tree in $G$).*

As a consequence of this theorem, we may solve the Steiner tree problem in the distance graph instead of the original graph; see Figure 5.1. One advantage is that we may assume that all Steiner points have degree at least 3, that the number of Steiner points is bounded by $n-2$, and that the tree has at most $2n-3$ edges. The Steiner tree problem for $D$ can be solved by enumerating all subsets $\overline{S} \subseteq V \setminus N_G$ of possible Steiner vertices, such that $|\overline{S}| \leq n-2$, and computing a *minimum spanning tree* in the subgraph of $D$ induced by $N_G \cup S$. This immediately leads to the following theorem:

**Theorem 5.2** *[187] The Steiner tree problem in a graph can be solved in polynomial-time in the following two cases:*

1. *The number of terminals $n = |N_G|$ is bounded by a constant.*

2. *The number of potential Steiner points $|V| - n$ is bounded by a constant.*

**Proof.** The number of Steiner vertex subsets in the distance graph is clearly bounded by $\min\{|V|^{n-2}, 2^{|V|-n}\}$. If $n$ is bounded by a constant, then the number of subsets is polynomial in $|V|$. If $|V| - n$ is bounded by a constant, then the number of subsets is also bounded by a constant. ∎

### 5.1.1 Graph reductions

For many problem instances of the Steiner tree problem, we can preprocess the input graph to reduce its size while preserving at least one minimum Steiner tree. For example, consider an edge $e = (u, v) \in E$ of weight $c(e)$. If there exists a path between $u$ and $v$ in $G$ of weight less than $c(e)$, then $e$ may clearly be disregarded.

Graph reductions play a central role for exact algorithms, since exact algorithms (in the worst case) have exponential running time behaviour in the size of the graph. In practice, some of the best graph reduction algorithms can reduce the original graph to a small fraction of its original size by using a (low-order) polynomial-time effort.

The first systematic studies of graph reductions were presented by Duin [141], Duin and Volgenant [143, 144], and Hwang, Richards and Winter [213]. Winter [406] proposed a series of reductions that were particularly powerful for grid graphs (for example, the Hanan grid graph). Duin [142] and Uchoa, Aragão and Ribeiro [376] extended the ideas of Winter further, resulting in powerful reductions for grid graphs coming from chip design. Currently, the fastest and most powerful graph reduction techniques are due to Polzin and Vahdati Daneshmand [127, 309, 311, 312].

Graph reduction tests can be divided into two main types: *exclusion tests* and *inclusion tests*. Both types of tests are performed on edges and/or Steiner vertices. The purpose of exclusion tests is to show that there exists a minimum Steiner tree that *does not use* the edge or Steiner vertex in question. The purpose of inclusion tests is to show that there exists a minimum Steiner tree that *does use* the edge or Steiner vertex in question. The tests are performed sequentially, so at any given stage in the reduction process there exists a minimum Steiner tree $T$ for $G$ in the reduced graph, such that $T$ does not use the excluded edges and Steiner vertices — and such that $T$ does use the included edges and Steiner vertices.

A range of techniques is used to design reduction tests. One of the powerful techniques is based on *bottleneck Steiner distances* (see Section 1.3.2 in Chapter 1). Consider the distance graph $D$ of $G$, and let $\overline{T}$ be a minimum spanning tree in the subgraph induced by the terminals $N_G$ in $D$. The bottleneck Steiner distance between two terminals $u$ and $v$ is the weight of the longest edge on the unique path between $u$ and $v$ in $\overline{T}$. If $e = (u, v)$ has weight $c(e)$ greater than the bottleneck Steiner distance, edge $e$ can be excluded. This test can be extended in many directions, including the case where $u$ and/or $v$ are Steiner vertices.

Another powerful technique is based on the idea of *expansion*. The basic idea is to assume that some edge $e = (u, v) \in E$ is part of a minimum Steiner tree — thus forming a subtree $T'$ of a minimum Steiner tree. The question is now if $T'$ actually can be expanded into a minimum Steiner tree for all terminals. Each of the edges incident to $u$ and $v$ is iteratively added to $T'$; if for each of these expansions it can be shown that the expanded tree $T''$ cannot be part of some minimum Steiner tree, edge $e$ can be excluded. By using (limited) backtrack search starting in $e$, wider expansions can be investigated; if all these expansions fail, edge $e$ can be excluded.

The tests described above are so-called *alternative-based* reductions: for exclusion tests it is shown that edges or Steiner vertices can be excluded since equally good or better alternatives exist in the remaining graph (and vice versa for inclusion tests). Polzin and Vahdati Daneshmand [311, 312] introduced the notion of *bound-based tests*. Let UB be the weight of some (heuristic) Steiner tree for $G$, and consider some Steiner vertex $u$. Assume that we can compute a lower bound LB on the weight of a Steiner tree that includes Steiner vertex $u$. If LB $>$ UB, then Steiner vertex $u$ can be excluded. Polzin and Vahdati Daneshmand showed that efficiently computable lower bounds can be obtained by using, for example, Voronoi regions for $G$ and dual ascent methods based on linear programming relaxations.

### 5.1.2 Dynamic programming

Some of the first non-trivial exact algorithms for the Steiner tree problem were based on dynamic programming. The idea of dynamic programming is to solve problems recursively; that is, to solve problems bottom-up by storing optimal solutions to subproblems, and combining these solutions into optimal solutions for larger subproblems. Dynamic programming algorithms are particularly effective for special cases, for example, when the number of terminals and/or Steiner points is small.

For the Steiner tree problem a natural dynamic programming approach is to compute minimum Steiner trees for non-empty terminal subsets $X \subseteq N_G$; let the weight of such a minimum Steiner tree for $X$ be $c(X)$. The base case, $X = \{u, v\}$ where $u, v \in N_G$, is easy: here $c(X)$ is simply the weight $c_P(u, v)$ of a shortest path between $u$ and $v$ in $G$. The challenge is to compute $c(X)$ for larger subsets of $N_G$, and ultimately to compute $c(N_G)$.

#### Dreyfus-Wagner algorithm

The Dreyfus-Wagner algorithm [132] is a clever implementation of the dynamic programming paradigm. (Levin [253] independently suggested a similar approach.) Consider some vertex $v \in V \setminus X$. Let $c(X \cup v)$ be the weight of a minimum Steiner tree for $X \cup v$, and let $c_v(X \cup v)$ be the weight of a minimum Steiner tree for $X \cup v$ where $v$ has degree 2 or more.

Consider a minimum Steiner tree $T_v$ for the set $X \cup v$ where $v$ has degree 2 or more. Clearly, we can decompose $T_v$ into two subtrees $T_v^1$ and $T_v^2$, such that $T_v^1$ is a minimum Steiner tree for $X' \cup v$ (where $X' \subset X$), and $T_v^2$ is a minimum Steiner tree for $(X \setminus X') \cup v$. Thus we have:

$$c_v(X \cup v) = \min_{\emptyset \subset X' \subset X} \{c(X' \cup v) + c((X \setminus X') \cup v)\}. \tag{5.1}$$

Now, consider a minimum Steiner tree $T$ for the set $X \cup v$ without any restrictions on the degree of $v$. If $v$ has degree 2 or more in $T$, then obviously $c(X \cup v) = c_v(X \cup v)$. So assume that $v$ has degree 1. Consider the unique path $p_T(v, w)$ in $T$ from $v$ to the first vertex $w$ in $T$ that either is a terminal or is a Steiner point of degree 3 or more. (Hence, all interior vertices of $p_T(v, w)$, if any, are Steiner vertices of degree 2.) If $w$ is a terminal, the weight of $T$ is equal to the weight of a shortest path from $v$ to $w$ plus the weight of a minimum Steiner tree for $X$ (since $w \in X$). If $w$ is a Steiner vertex of degree 3 or more, the weight of $T$ is equal to the weight of a shortest path from $v$ to $w$ plus the weight of a minimum Steiner tree for $X \cup w$, where $w$ has degree 2 or more. Therefore, we have:

$$c(X \cup v) = \min\{\min_{w \in X}\{c_P(v, w) + c(X)\}, \min_{w \in V \setminus X}\{c_P(v, w) + c_w(X \cup w)\}\}. \tag{5.2}$$

The running time of the Dreyfus-Wagner algorithm is $O^*(3^n)$, where the function $O^*()$ ignores polynomial factors (see Exercise 5.2).

**Theorem 5.3** *[132]* ***[Dreyfus-Wagner algorithm]*** *The Steiner tree problem in a graph can be solved using dynamic programming in $O^*(3^n)$ time, where $n$ is the number of terminals.*

For small terminal sets, the Dreyfus-Wagner algorithm is much more efficient than the spanning tree enumeration algorithm (see Theorem 5.2), and it is still the method of choice today for problem instances with small terminal sets. Although theoretically faster algorithms exist (with a running time of $O^*(c^n)$ for any $c > 2$ [159, 381]), it is not clear if these algorithms are faster in practice due to the large constants involved. Efficient variants of the Dreyfus-Wagner algorithm can solve large problem instances to optimality [205].

### 5.1.3 Integer programming

Like many other combinatorial optimisation problems, the Steiner tree problem in graphs can be modelled using (linear) integer programming. The use of integer programming formulations for the problem dates back to Yang and Wing [425], who formulated a Steiner tree packing problem using an integer programming model. Aneja [15] and Wong [411] pioneered the use of integer programming for the Steiner tree problem in graphs in the early 1980s.

In this section we present four integer programming formulations for the Steiner tree problem in graphs. Our first formulation, called the *spanning tree* formulation, has not been used much for the graph problem, but it has successfully been used to compute minimum Steiner trees and minimum spanning trees in *hypergraphs* (see Section 5.2). The second formulation is Aneja's *cut* formulation, and the third formulation is the one that is used by current state-of-the-art exact algorithms — namely Wong's *directed cut* formulation. Finally, we present a *multi-commodity flow* formulation that has polynomially many variables and constraints.

## Spanning tree formulation

The main idea of this formulation is to keep track of the Steiner vertices that are part of the Steiner tree. When the set of Steiner vertices is known, the problem reduces to a minimum spanning tree problem on the terminals and the selected Steiner vertices. Thus, well-known integer programming formulations for the minimum spanning tree problem can be used.

We represent a Steiner tree as an incidence vector $x$, where $x_e = 1$ if edge $e \in E$ is part of the Steiner tree, and otherwise $x_e = 0$. Clearly, the weight of a Steiner tree is then $\sum_{e \in E} c(e)x_e$. For any edge set $F \subseteq E$ define $x(F) = \sum_{e \in F} x_e$ to be the sum of $x$ over edge set $F$ (in other words, the number of edges of the Steiner tree in $F$).

Let $S_G = V \setminus N_G$ be the set of candidate Steiner vertices in $G$. Define an incidence vector $y$, such that $y_v = 1$ if $v \in S_G$ is part of the Steiner tree, and otherwise $y_v = 0$. For any vertex set $W \subseteq V$ define $y(W) = \sum_{v \in W} y_v$ to be the sum of $y$ over vertex set $W$; denote by $E(W) \subseteq E$ the set of edges with both endpoints in $W$. The spanning tree formulation $IP_{\text{spt}}$ is as follows:

$$\begin{array}{llll}
(5.3) & \text{minimise} & \sum\limits_{e \in E} c(e)x_e & \boxed{IP_{\text{spt}}} \\
(5.4) & \text{subject to} & x(E) = y(S_G) + |N_G| - 1 & \\
(5.5) & & x(E(W)) \le y(S_G \cap W) + |N_G \cap W| - 1, & \begin{array}{r} W \subset V, \\ W \cap N_G \neq \emptyset \end{array} \\
(5.6) & & x_e \in \{0,1\}, & e \in E \\
(5.7) & & y_v \in \{0,1\}, & v \in S_G.
\end{array}$$

This formulation was suggested and studied by Goemans and Myung [183], and it has its origin in Edmond's description of the spanning tree polytope [148]. The objective function (5.3) is the sum of the weights of the selected edges. Equality (5.4) states that the number of edges in a spanning tree is equal to the number of vertices minus 1. Finally, the so-called *generalised subtour elimination* constraints (5.5) state that the number of tree edges within any vertex subset $W$ should be at most the number of tree vertices minus 1 — otherwise a cycle (or subtour) would be created. The number of generalised subtour elimination constraints is exponential, so the LP-relaxation is solved iteratively by adding violated constraints; the separation problem (finding a linear inequality that separates the optimum point from the convex hull of the true feasible set) is equivalent to a minimum cut problem that can be solved in polynomial time (see Section 5.2).

## Cut formulation

In the cut formulation [15] the focus is on the connectivity of the Steiner tree — namely that there should be path between any pair of terminals in the Steiner tree.

For a set of vertices $W \subseteq V$, let $\delta(W) \subseteq E$ denote the *cut* given by $W$, that is, the set of edges with exactly *one* endpoint in $W$. Choose any terminal vertex $r \in N_G$ as the *root* of the Steiner tree. The cut formulation $IP_{\text{cut}}$ is as follows:

$$\begin{array}{llll} (5.8) & \text{minimise} & \sum_{e \in E} c(e) x_e & \boxed{IP_{\text{cut}}} \\ (5.9) & \text{subject to} & x(\delta(W)) \geq 1, & W \subseteq V \setminus \{r\},\ W \cup N_G \neq \emptyset \\ (5.10) & & x_e \in \{0,1\}, & e \in E. \end{array}$$

The *cut* constraints (5.9) ensure that for any cut that separates $r$ from some other terminal, there is at least one edge that crosses the cut. Clearly, the resulting Steiner tree is feasible, as it connects $r$ to each of the remaining terminals. Again, the number of cut constraints is exponential. The separation problem can be solved in polynomial time by solving at most $|N_G| - 1$ minimum cut problems (as described in the next section).

The cut formulation is sometimes denoted the *set cover* formulation [213], since it is actually a special case of the set cover problem: each cut in the graph that separates the root from some other terminal needs to be covered by at least one edge in the Steiner tree.

## Directed cut formulation

The directed cut formulation [411] is a seemingly minor variant of the cut formulation, but it turns out that the LP-relaxation is much stronger.

Consider an edge-weighted directed graph $D = (V, A)$ constructed from $G = (V, E)$. The directed graph $D$ has the same vertex set as $G$, and for each edge $(u, v) \in E$, there are two opposite arcs $[u, v]$ and $[v, u]$ in $A$; the weight $c(a)$ of an arc $a \in A$ is the same as the weight $c(e)$ of the underlying edge $e \in E$. A *Steiner arborescence* in $D$ is a directed tree rooted in some terminal $r \in N_G$ that spans all the terminals $N_G$. Clearly, a minimum-weight Steiner arborescence in $D$ corresponds to a minimum Steiner tree in $G$.

We represent a Steiner arborescence as an incidence vector $w$, where $w_a = 1$ if arc $a \in A$ is part of the Steiner arborescence, and otherwise $w_a = 0$. Clearly, the weight of a Steiner arborescence is then $\sum_{a \in A} c(a)w_a$. For any arc set $B \subseteq A$ define $w(B) = \sum_{a \in B} w_a$ to be the sum of $w$ over arc set $B$. For a set of vertices $W \subseteq V$, let $\delta^-(W) = \{[u, v] \in A : u \notin W, v \in W\}$ denote the *directed cut* given by $W$, that is, the set of arcs ending (but not beginning) in $W$. The directed cut formulation $IP_{\text{dicut}}$ is as follows:

$$\begin{array}{llll}
(5.11) & \text{minimise} & \displaystyle\sum_{a \in A} c(a)w_a & \boxed{IP_{\text{dicut}}} \\
(5.12) & \text{subject to} & w(\delta^-(W)) \geq 1, & W \subseteq V \setminus \{r\},\ W \cup N_G \neq \emptyset \\
(5.13) & & w_a \in \{0, 1\}, & a \in A.
\end{array}$$

As before, the *directed cut* constraints (5.12) ensure that for any cut that separates $r$ from some other terminal, there is at least one edge that crosses the cut. The separation problem can be solved as follows. Consider some fractional solution $w_a$, $a \in A$. Set up a flow network on $A$ with $r$ as source, one of the terminals $t \in N_G \setminus \{r\}$ as sink, and $w_a$ as the (fractional) capacity of arc $a \in S$. Compute a minimum-capacity cut in this network. If the minimum cut has capacity less than 1, this cut corresponds to a violated directed cut constraint.

## Multi-commodity flow formulation

A formulation that uses a polynomial number of constraints can be obtained by setting up a multi-commodity flow network. Consider again the directed graph $D = (V, A)$ from the directed cut formulation. As before, let $\delta^-(v)$ denote the set of arcs ending in a vertex $v \in V$. Similarly, let $\delta^+(v)$ denote the set of arcs beginning in $v \in V$.

For each terminal $t \in N_G \setminus \{r\}$, we would like to send one unit of flow from $r$ to $t$. Let $f^t$ be the flow vector defining the flow from $r$ to $t$, such that $f^t_a$ is the flow on arc $a \in A$. For any arc set $B \subseteq A$ define $f^t(B) = \sum_{a \in B} f^t_a$ to be the sum of $f^t$ over arc set $B$. The multi-commodity flow formulation $IP_{\text{mcf}}$ is as follows:

(5.14)

$$\text{minimise} \quad \sum_{a \in A} c(a) w_a \qquad \boxed{IP_{\text{mcf}}}$$

(5.15)

$$\text{subject to} \quad f^t(\delta^+(v)) - f^t(\delta^-(v)) = \begin{cases} 1, & v = r \\ -1, & v = t \\ 0, & v \in N_G \setminus \{r, t\} \end{cases}, \qquad \begin{array}{l} v \in V, \\ t \in N_G \setminus \{r\} \end{array}$$

(5.16) $$f_a^t \le w_a, \qquad \begin{array}{l} a \in A, \\ t \in N_G \setminus \{r\} \end{array}$$

(5.17) $$w_a \in \{0, 1\}, \qquad a \in A$$

(5.18) $$f_a^t \ge 0, \qquad \begin{array}{l} a \in A, \\ t \in N_G \setminus \{r\} \end{array}.$$

The flow conservation constraints (5.15) ensure that $f_t$ corresponds to one unit of flow from $r$ to $t$. The necessary capacity $w_a$ on arc $a \in A$ is given by constraints (5.16). Clearly, the number of constraints is bounded by $O(|V| \max\{|V|, |A|\})$, or polynomial in the size of the directed graph $D = (V, A)$.

### Relation between optimal solutions to LP-relaxations

How are the optimal values of the four LP-relaxations $LP_{\text{spt}}$, $LP_{\text{cut}}$, $LP_{\text{dicut}}$ and $LP_{\text{mcf}}$ related? For each formulation the optimal value of the LP-relaxation is independent of the choice of the root $r \in N_G$ [183]. Denote by $v(LP_{\text{spt}})$, $v(LP_{\text{cut}})$, $v(LP_{\text{dicut}})$ and $v(LP_{\text{mcf}})$ the optimal values of the four LP-relaxations for a given problem instance.

**Lemma 5.4** *[99, 183]*

$$v(LP_{\textit{cut}}) \le v(LP_{\textit{dicut}}) = v(LP_{\textit{spt}}) = v(LP_{\textit{mcf}}).$$

The relation $v(LP_{\text{cut}}) \le v(LP_{\text{dicut}})$ was shown by Chopra and Rao [99], who also proved that even if the cut formulation is strengthened using so-called Steiner partition inequalities and odd hole inequalities, the directed cut formulation remains stronger than the cut formulation. In practice $v(LP_{\text{cut}})$ is significantly worse than $v(LP_{\text{dicut}})$, so even if the LP-relaxation $LP_{\text{dicut}}$ has twice as many variables as $LP_{\text{cut}}$, it is more than worth the extra effort.

The relation $v(LP_{\text{dicut}}) = v(LP_{\text{spt}})$ essentially follows from the fact that we can transform any solution to $LP_{\text{dicut}}$ to a solution for $LP_{\text{spt}}$ — and vice versa. Consider a solution $w$ to $LP_{\text{dicut}}$; define $x_{(u,v)} = w_{[u,v]} + w_{[v,u]}$, $(u, v) \in E$, and $y_v = w(\delta^-(v))$, $v \in S_G$. Then $(x, y)$ is a feasible solution to $LP_{\text{spt}}$ (see Exercise 5.3). The other direction is similar, but a bit more involved [183].

The relation $v(LP_{\text{dicut}}) = v(LP_{\text{mcf}})$ follows from the max-flow min-cut theorem. Any feasible solution to $LP_{\text{dicut}}$ corresponds to a feasible solution to $LP_{\text{mcf}}$, since each relevant cut has capacity at least 1. Also, any feasible solution to $LP_{\text{mcf}}$ corresponds to a feasible solution to $LP_{\text{dicut}}$, since the flow across any relevant cut is at least 1 [183].

The directed cut formulation has been the preferred formulation for all recent exact algorithms for the Steiner tree in graphs [98, 127, 236, 309, 311, 314]. The LP-relaxation can be approximated very efficiently using dual ascent methods [311, 411].

### Other integer programming formulations

A number of other integer programming formulations exist for the Steiner tree problem in graphs. For example, the problem can be formulated as a degree-constrained minimum spanning tree problem. Also, a number of facet-defining inequalities can be added to strengthen these formulations. For an overview of formulations and polyhedral properties, see [99, 100, 213, 183, 310, 313, 314]. Integer programming formulations for a wider range of tree problems on graphs are studied in [181, 182, 273].

## 5.2 Spanning trees and Steiner trees in hypergraphs

In this section we consider a generalisation of (undirected) graphs denoted *hypergraphs*. In hypergraphs, edges can connect two or more vertices.

**Definitions [Hyperedge, hypergraph]**: Given a finite set $V$, a *hyperedge* $\mathbf{e} \subseteq V$ is a subset of $V$ of cardinality $|\mathbf{e}| \geq 2$. A *hypergraph* $\mathbf{G} = (V, \mathbf{E})$ consists of a set of vertices $V$ and a set $\mathbf{E}$ of hyperedges of $V$.

An example of a hypergraph is given in Figure 5.2 (left).

Two problems are discussed in this section. First we consider a generalisation of the minimum spanning tree problem in graphs to hypergraphs. It turns out that this generalisation makes the problem NP-hard — in contrast with the minimum spanning tree problem in graphs, which can be solved in polynomial time. The second problem is a generalisation of the Steiner tree problem in graphs to hypergraphs.

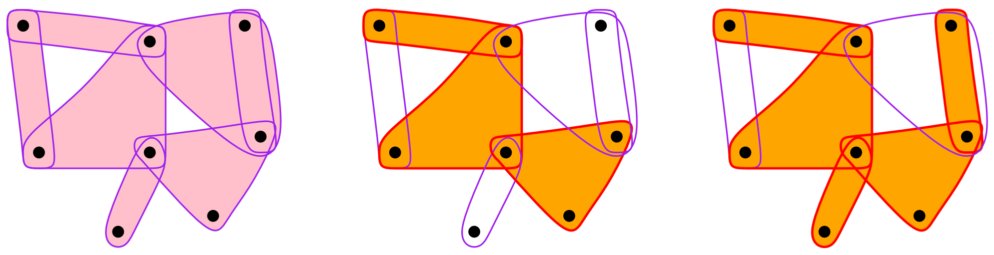


Figure 5.2: An example of a hypergraph $\mathbf{G}$ containing 8 vertices and 7 hyperedges. The middle diagram shows a chain in $\mathbf{G}$, while the diagram on the right shows a spanning tree for $\mathbf{G}$.

### 5.2.1 Spanning trees in hypergraphs

Below we give a formal definition of the minimum spanning tree problem in hypergraphs, and then we discuss a number of exact algorithms for the problem. First we generalise the concept of paths and (spanning) trees to hypergraphs.

**Definition [Chain in hypergraph]**: A *chain* in a hypergraph $\mathbf{G} = (V, \mathbf{E})$ from $v_0 \in V$ to $v_k \in V$ is an alternating sequence of vertices and hyperedges $v_0, \mathbf{e}_1, v_1, \mathbf{e}_2, v_2, \ldots, \mathbf{e}_k, v_k$ such that all vertices and hyperedges are *distinct* and $v_{i-1}, v_i \in \mathbf{e}_i$ for $i = 1, 2, \ldots, k$.

**Definition [Tree in hypergraph, spanning tree in hypergraph]**: A *tree* $\mathbf{T} = (V(\mathbf{T}), \mathbf{E}(\mathbf{T}))$ in a hypergraph $\mathbf{G} = (V, \mathbf{E})$ is a set of hyperedges $\mathbf{E}(\mathbf{T}) \subseteq \mathbf{E}$ spanning a set of vertices $V(\mathbf{T}) = \cup_{\mathbf{e}\in\mathbf{E}(\mathbf{T})}\mathbf{e}$ such that there is a *unique* chain using vertices and hyperedges from $\mathbf{E}(\mathbf{T})$ between *every pair* of vertices $u, v \in V(\mathbf{T})$. A *spanning tree* $\mathbf{T} = (V(\mathbf{T}), \mathbf{E}(\mathbf{T}))$ in a hypergraph $\mathbf{G} = (V, \mathbf{E})$ has $V(\mathbf{T}) = V$.

These concepts are illustrated in Figure 5.2. Note that the uniqueness of chains in a tree in a hypergraph implies that any two distinct hyperedges in a tree contain at most one common vertex. Furthermore, a (spanning) tree in a hypergraph has the same key property as a (spanning) tree in an ordinary graph — namely that there is a unique path in the tree between every pair of vertices spanned by the tree.

MINIMUM SPANNING TREE PROBLEM IN HYPERGRAPHS
**Given**: An edge-weighted hypergraph $\mathbf{G} = (V, \mathbf{E})$, where $c(\mathbf{e}) > 0$ denotes the weight of each hyperedge $\mathbf{e} \in \mathbf{E}$.
**Find**: A spanning tree $\mathbf{T} = (V, \mathbf{E}(\mathbf{T}))$ in $\mathbf{G}$ such that $\sum_{\mathbf{e}\in\mathbf{E}(\mathbf{T})} c(\mathbf{e})$ is minimised.

The minimum spanning tree problem in hypergraphs is (in general) NP-hard when the hypergraph contains edges of cardinality $4$ or more [318, 388].[59] The main motivation for studying the minimum spanning tree problem in hypergraphs is that it can be used to solve the full Steiner tree (FST) concatenation problem (see Section 1.4.4 in Chapter 1). Recall that the output of the FST generation phase is a set of FSTs $\mathcal{F} = \{T_1, T_2, \ldots, T_m\}$ that is guaranteed to contain the full components of at least one minimum Steiner tree for terminal set $N$. In the FST concatenation problem, we need to identify a subset $\mathcal{F}^* \subseteq \mathcal{F}$ such that $\mathcal{F}^*$ interconnects $N$ and has minimum total length. Clearly, the FST concatenation problem can be formulated as a minimum spanning tree problem in a hypergraph, where the terminal set $N$ forms the vertex set; for each FST $T_i$ the associated terminal subset $N(T_i) \subseteq N$ is a hyperedge with weight equal to the geometric length $|T_i|$ of the FST.

It should be noted that the pure minimum spanning tree problem in hypergraphs does not capture the underlying geometry of the associated FSTs. For example, based on the geometry of the FSTs, it may be known that certain pairs of FSTs cannot appear together in a minimum Steiner tree, for example, by using the notion of FST compatibility [113, 409]. Also, so-called FST pruning methods (or hypergraph reduction methods) can be used to eliminate FSTs from consideration or to identify FSTs that must be in a minimum Steiner tree [113, 155, 409].

### Backtrack search

Backtrack search is a simple, yet reasonably effective algorithm for the minimum spanning tree problem in a hypergraph $\mathbf{G} = (V, \mathbf{E})$. Starting with a partial solution consisting of a single hyperedge $\mathbf{e}_1 \in \mathbf{E}$, we seek a minimum spanning tree containing $\mathbf{e}_1$. This is done by recursively adding hyperedges to the solution so that it remains connected — until it spans $V$ or it can be concluded that it cannot be optimal, for example, if a cycle is created. In this case the search backtracks and some other hyperedges are added. Obviously, it is only necessary to try hyperedges spanning a particular vertex as the initial hyperedge.

The cut-off tests (for non-optimality) applied during the search determine the practical behaviour of the algorithm. Winter's original GeoSteiner implementation from 1985 used only relatively simple cut-off tests, and he observed that the backtrack search algorithm for FST concatenation began to dominate the FST generation algorithm already at 15 terminals [404]. Over the next decade a number of increasingly sophisticated techniques were applied to reduce the running time of backtrack search [112, 113, 335, 409], allowing the solution of FST concatenation problems with up to around 150 terminals.

**Dynamic programming**

Ganley and Cohoon [162, 163, 165] designed the first dynamic programming algorithm for the minimum spanning tree problem in a hypergraph. The algorithm was originally stated as an algorithm to solve the FST concatenation problem for the rectilinear Steiner tree problem in the plane, and it was called *full set dynamic programming*.

The algorithm computes the weight $c(X)$ of a minimum spanning tree for every non-empty subset $X \subseteq V$ (or decides that no such tree exists). Subsets $X$ of $V$ are enumerated in order of increasing cardinality starting with subsets of size 2.

Consider a subset $X$. Let $\mathbf{E}(X) = \{\mathbf{e} \in \mathbf{E} : \mathbf{e} \subseteq X\}$ be the set of edges that are completely contained in $X$. The main observation is that a minimum spanning tree for $X$ either consists of a single hyperedge $\mathbf{e} \in \mathbf{E}(X)$ where $\mathbf{e} = X$, or consists of a hyperedge $\mathbf{e} \in \mathbf{E}(X)$ joined with a minimum spanning tree for $v \cup (X \setminus \mathbf{e})$, where $v \in \mathbf{e}$. Formally we have:

$$(5.19) \qquad c(X) = \min_{\mathbf{e} \in \mathbf{E}(X)} \begin{cases} c(\mathbf{e}), & \mathbf{e} = X \\ \min_{v \in \mathbf{e}} \{c(\mathbf{e}) + c(v \cup (X \setminus \mathbf{e}))\}, & \mathbf{e} \subset X \end{cases}.$$

For $n = |V|$ the running time of this dynamic programming algorithm is $O^*((1+\phi)^n)$, when the number of hyperedges is bounded by $O(\phi^n)$. When the number of hyperedges is polynomial in $n$, the running time drops to $O^*(2^n)$ (see Exercise 5.4).

Ganley and Cohoon applied this dynamic programming algorithm to the rectilinear Steiner tree problem in the plane, where they obtained a worst-case running time of $O^*(2.62^n)$. This running time has been improved to $O^*(2.38^n)$ by Fößmeier and Kaufmann [154], and more recently to $O^*(2.357^n)$ by Fuchs, Kern and Wang [160].

Although dynamic programming provides the best worst-case bounds for the mimimum spanning tree problem in hypergraphs, the practical behaviour seems to be inferior to backtrack search [154]. In addition, huge memory requirements make the approach impractical for problem instances with more than 40 vertices.

**Integer programming**

The first integer programming formulation for the minimum spanning tree problem in hypergraphs was proposed by Warme [388] in 1998. Warme generalised the *spanning tree* formulation for graphs (see Section 5.1.3) to hypergraphs. This approach turned out to be a major breakthrough for solving the minimum spanning tree problem in hypergraphs — and hence the FST concatenation problem — allowing the solution of problems of several orders of magnitude greater than was possible using either backtrack search or dynamic programming.

A minimum spanning tree is represented as an incidence vector $x$, where $x_{\mathbf{e}} = 1$ if edge $\mathbf{e} \in \mathbf{E}$ is part of the minimum spanning tree, and otherwise $x_{\mathbf{e}} = 0$. Recall that $|\mathbf{e}|$ denotes the number of vertices spanned by hyperedge $\mathbf{e} \in \mathbf{E}$. The spanning tree formulation $IP_{\text{hmspt}}$ is as follows:

$$\boxed{IP_{\text{hmspt}}}$$

$$\begin{aligned}
&(5.20) && \text{minimise} && \sum_{\mathbf{e}\in\mathbf{E}} c(\mathbf{e})x_{\mathbf{e}} && \\
&(5.21) && \text{subject to} && \sum_{\mathbf{e}\in\mathbf{E}} (|\mathbf{e}|-1)x_{\mathbf{e}} = |V|-1 && \\
&(5.22) && && \sum_{\substack{\mathbf{e}\in\mathbf{E}:\\ \mathbf{e}\cap W\neq\emptyset}} (|\mathbf{e}\cap W|-1)x_{\mathbf{e}} \le |W|-1, && \emptyset \neq W \subset V \\
&(5.23) && && x_{\mathbf{e}} \in \{0,1\}, && \mathbf{e}\in\mathbf{E}.
\end{aligned}$$

The objective (5.20) is to minimise the total weight of the chosen hyperedges subject to the following constraints: Equation (5.21) enforces the correct number and cardinality of hyperedges to construct a spanning tree. The intuition behind this equation is that the number of 2-edges in a spanning tree of an ordinary graph is one less than the number of vertices. We can think of the number of 2-edges in a hyperedge $\mathbf{e} \in \mathbf{E}$ as being $|\mathbf{e}| - 1$, corresponding to a local tree of 2-edges interconnecting the vertices of the hyperedge.

Constraints (5.22) eliminate cycles by extending the standard notion of subtour elimination constraints. For a given non-empty subset $W \subset V$, the total number of 2-edges in the subset (again viewing hyperedges as a set of 2-edges) can be at most $|W|-1$, otherwise a cycle is created. The number of 2-edges contributed by a hyperedge $\mathbf{e} \in \mathbf{E}$ that intersects $W$ is $|\mathbf{e} \cap W| - 1$.

Warme [388] proved several fundamental properties of the polytope corresponding to these constraints, including the fact that the subtour elimination constraints (5.22) are facet-defining for $|V| \geq 3$. This result contributes to explaining the strength of the formulation.

The integer program is solved via branch-and-cut. Lower bounds are provided by LP-relaxation. The number of subtour elimination constraints (5.22) is exponential in $|V|$, and the constraints are therefore dynamically added by separation methods. The separation problem can be solved in polynomial time (in $|V|$ and $|\mathbf{E}|$) by solving a series of minimum cut problems [388]. However, heuristic separation methods are also used whenever applicable in order to speed up convergence to LP-optimum.

Alternative integer programming formulations for the minimum spanning tree problem in hypergraphs were studied by Polzin and Vahdati Daneshmand [313, 314]. They proved that a directed version of the spanning tree formulation of Warme is equivalent to Warme's formulation — in the sense that the optimal solutions to the LP-relaxations are identical. Also, a hypergraph version of the *directed cut* formulation for graphs (see Section 5.1.3) is equivalent to the spanning tree formulation. Finally, Polzin and Vahdati Daneshmand presented examples where the hypergraph formulation is strictly stronger than the graph version of the directed cut formulation for the FST concatenation problem; however, alternative formulations of the graph problem were shown to be incomparable to the hypergraph formulation.

### 5.2.2 Steiner trees in hypergraphs

Consider the following generalisation of the minimum spanning tree problem in hypergraphs:

STEINER TREE PROBLEM IN HYPERGRAPHS
**Given**: An edge-weighted hypergraph $\mathbf{G} = (V, \mathbf{E})$, where $c(\mathbf{e}) > 0$ denotes the weight of each hyperedge $\mathbf{e} \in \mathbf{E}$, and a set of terminals $N_G \subseteq V$.
**Find**: A tree $\mathbf{T} = (V(\mathbf{T}), \mathbf{E}(\mathbf{T}))$ in $\mathbf{G}$ such that $N_G \subseteq V(\mathbf{T})$, and such that $\sum_{\mathbf{e}\in\mathbf{E}(\mathbf{T})} c(\mathbf{e})$ is minimised.

The motivation for studying this problem comes from the FST concatenation problem when considering the *obstacle-avoiding* Steiner tree problem in the plane (see Section 4.2 in Chapter 4). The problem was first formulated by Zachariasen and Winter [435] in 1999. In the obstacle-avoiding problem, the vertices $V \setminus N_G$ correspond to obstacle corners, and the FSTs span both terminals and obstacle corners. In the minimum Steiner tree, obstacle corners may or may not be included.

The Steiner tree problem in hypergraphs can be solved using both backtrack search and dynamic programming, as described in the previous section. The problem can also be formulated as an integer programming using the spanning tree formulation. As before, we represent a Steiner tree as an incidence vector $x$, where $x_{\mathbf{e}} = 1$ if edge $\mathbf{e} \in \mathbf{E}$ is part of the Steiner tree, and otherwise $x_{\mathbf{e}} = 0$. Let $S_G = V \setminus N_G$ be the set of Steiner vertices in $\mathbf{G}$. Define incidence vector $y$, where $y_v = 1$ if Steiner vertex $v \in S_G$ is part of the Steiner tree, and otherwise $y_v = 0$. The spanning tree formulation $IP_{\text{hsspt}}$ is as follows:

(5.24)

$$\text{minimise} \quad \sum_{\mathbf{e}\in\mathbf{E}} c(\mathbf{e})x_{\mathbf{e}} \qquad \boxed{IP_{\text{hsspt}}}$$

(5.25)

$$\text{subject to} \quad \sum_{\mathbf{e}\in\mathbf{E}}(|\mathbf{e}|-1)x_{\mathbf{e}} = \sum_{v\in S_G} y_v + |N_G| - 1$$

(5.26)

$$\sum_{\substack{\mathbf{e}\in\mathbf{E}:\\ \mathbf{e}\cap W\neq\emptyset}} (|\mathbf{e}\cap W|-1)x_{\mathbf{e}} \le \sum_{v\in S_G\cap W} y_v + |N_G\cap W| - 1, \qquad \begin{array}{r} W\subset V,\\ W\cap N_G\neq\emptyset\end{array}$$

(5.27)

$$x_{\mathbf{e}} \in \{0,1\}, \qquad \mathbf{e}\in\mathbf{E}$$

(5.28)

$$y_v \in \{0,1\}, \qquad v\in S_G.$$

Equation (5.25) enforces the correct number and cardinality of hyperedges to construct a spanning tree given that we have chosen $\sum_{v\in S_G} y_v$ Steiner vertices. Constraints (5.26) eliminate cycles using subtour elimination constraints; the right-hand side counts the total number of vertices chosen within subset $W$ (minus 1). As for the minimum spanning tree problem, the separation problem for constraints (5.26) can be solved in polynomial time (in $|V|$ and $|\mathbf{E}|$) by solving a series of minimum cut problems [388, 435]. Valid inequalities based on the underlying geometric problem, such as bounds on the degrees of terminals and Steiner vertices, can be added to the formulation [206, 435].

## Exercises

**5.1.** Show that the Steiner tree problem in graphs with zero-weight and/or negative-weight edges reduces to the Steiner tree problem with positive edge weights.

**5.2.** Show that the running time of the Dreyfus-Wagner algorithm is $O^*(3^n)$, where $n$ is the number of terminals. [Hint: Show that a terminal can appear in three different sets in recursion (5.1) and in two different sets in recursion (5.2).]

**5.3.** Consider a solution $w$ to $LP_{\text{dicut}}$; define $x_{(u,v)} = w_{[u,v]} + w_{[v,u]}$, $(u,v)\in E$, and $y_v = w(\delta^-(v))$, $v\in S_G$. Show that $(x,y)$ is a feasible solution to $LP_{\text{spt}}$.

**5.4.** Show directly using recursion (5.19) that the running time of the dynamic programming algorithm for the minimum spanning tree problem in hypergraphs becomes $O^*(2^n)$, when the number of vertices is $n$ and the number of hyperedges is polynomial in $n$.

## Notes

[58]The Steiner tree problem in graphs was originally formulated by Hakimi [187] (and independently by Levin [253]) in 1971. In the literature the problem is sometimes called the *Steiner problem in networks* (and the graph version is reserved for the unweighted case). Exact algorithms based on enumeration and dynamic programming were first proposed by Hakimi [187], Levin [253] and Dreyfus and Wagner [132].

[59]For hypergraphs with edges of size 3, there exists a polynomial-time algorithm for the *unweighted* case [318, 271]. Note that for a different definition of the minimum spanning tree problem in a hypergraph, Tomescu and Zimand [372] have shown that the problem is NP-hard for hypergraphs with edges of size 3.

# Bibliography

[1] N. Abboud, M. Grötschel, and T. Koch. Mathematical methods for physical layout of printed circuit boards: an overview. *OR Spectrum*, 30(3):453–468, 2008.

[2] M. Abellanas, F. Hurtado, C. Icking, R. Klein, E. Langetepe, L. Ma, B. Palop, and V. Sacristán. The farthest color Voronoi diagram and related problems. Technical report, Department of Computer Science I, University of Bonn, 2006.

[3] A. R. Agnihotri and P. H. Madden. Congestion reduction in traditional and new routing architectures. In *Proceedings of the 13th ACM Great Lakes Symposium on VLSI (GLSVLSI)*, pages 211–214, Washington, DC, 2003.

[4] A. V. Aho, M. R. Garey, and F. K. Hwang. Rectilinear Steiner trees: Efficient special-case algorithms. *Networks*, 7(1):37–58, 1977.

[5] G. Ajwani, C. Chu, and W.-K. Mak. FOARS: FLUTE based obstacle-avoiding rectilinear Steiner tree construction. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 30(2):194–204, 2011.

[6] J. Akiyama, X. Chen, G. Nakamura, and M. Ruiz. Minimum perimeter developments of the platonic solids. *Thai Journal of Mathematics*, 9(3):461–487, 2012.

[7] M. Alfaro, M. Conger, K. Hodges, A. Levy, R. Kochar, L. Kuklinski, Z. Mahmood, and K. von Haam. The structure of singularities in $\phi$-minimizing networks in $\mathbf{R}^2$. *Pacific Journal of Mathematics*, 149:201–210, 1991.

[8] C. Alford, M. Brazil, and D. H. Lee. Optimisation in underground mining. In *Handbook of Operations Research in Natural Resources*, pages 561–577. Springer, New York, 2007.

[9] C. J. Alpert, T. C. Hu, J. H. Huang, and A. B. Kahng. A direct combination of the Prim and Dijkstra constructions for improved performance-driven global routing. In *Proceedings of the IEEE International Symposium on Circuits and Systems*, pages 1868–1872, Chicago, Illinois, 1993.

[10] C. J. Alpert, Z. Li, M. D. Moffitt, G.-J. Nam, J. A. Roy, and G. Tellez. What makes a design difficult to route. In *Proceedings of the 19th ACM International Symposium on Physical Design (ISPD)*, pages 7–12, New York, NY, 2010.

[11] C. J. Alpert, D. P. Mehta, and S. S. Sapatnekar, editors. *Handbook of Algorithms for Physical Design Automation*. CRC Press, London, 2009.

[12] E. Althaus. Berechnung optimaler Steinerbäume in der Ebene. Master's thesis, Max-Planck-Institut für Informatik in Saarbrücken, Universität des Saarlandes, 1998.

[13] E. Althaus, J. Kupilas, and R. Naujoks. On the low-dimensional Steiner minimum tree problem in Hamming metric. *Theoretical Computer Science*, 505:2–10, 2013.

[14] E. Althaus and R. Naujoks. Computing Steiner minimum trees in Hamming metric. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm (SODA)*, pages 172–181, Miami, Florida, 2006.

[15] Y. P. Aneja. An integer linear programming approach to the Steiner problem in graphs. *Networks*, 10:167–178, 1980.

[16] S. Arora. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *Journal of the ACM*, 45(5):753–782, 1998.

[17] T. Asano, T. Asano, L. Guibas, J. Herchberger, and H. Imai. Visibility of disjoint polygons. *Algorithmica*, 1:49–63, 1986.

[18] F. Aurenhammer and R. Klein. Voronoi diagrams. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, chapter 5, pages 201–290. North-Holland, Amsterdam, 2000.

[19] M. Avci and S. Yamacli. An improved Elmore delay model for VLSI interconnects. *Mathematical and Computer Modelling*, 51(7):908–914, 2010.

[20] B. Awerbuch, A. Baratz, and D. Peleg. Cost-sensitive analysis of communication protocols. In *Proceedings of the Ninth Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 177–187, New York, NY, 1990.

[21] S. W. Bae, S. Choi, C. Lee, and S. Tanigawa. Exact algorithms for the bottleneck Steiner tree problem. *Algorithmica*, 61(4):924–948, 2011.

[22] S. W. Bae, C. Lee, and S. Choi. On exact solutions to the Euclidean bottleneck Steiner tree problem. *Information Processing Letters*, 110(16):672–678, 2010.

[23] S. Bainbridge, D. Eggeling, and G. Page. Lessons from the field - Two years of deploying operational wireless sensor networks on the Great Barrier Reef. *Sensors*, 11(7):6842–6855, 2011.

[24] C. Bartoschek, S. Held, J. Maßberg, D. Rautenbach, and J. Vygen. The repeater tree construction problem. *Information Processing Letters*, 110(24):1079–1083, 2010.

[25] J. Beardwoord, J. H. Halton, and J. M. Hammersley. The shortest path through many points. *Mathematical Proceedings of the Cambridge Philosophical Society*, 55(4):299–327, 1959.

[26] B. Berger, M. L. Brady, D. J. Brown, and T. Leighton. Nearly optimal algorithms and bounds for multilayer channel routing. *Journal of the ACM*, 42:500–542, 1995.

[27] P. Berman and V. Ramaiyer. Improved approximations for the Steiner tree problem. *Journal of Algorithms*, 17(3):381–408, 1994.

[28] P. Berman and A. Zelikovsky. On approximation of the power-$p$ and bottleneck Steiner trees. In D.-Z. Du, J .M. Smith, and J. H. Rubinstein, editors, *Advances in Steiner Trees*, volume 6 of *Combinatorial Optimization*, pages 117–135. Springer, New York, 2000.

[29] M. Bern and D. Bienstock. Polynomially solvable special cases of the Steiner problem in planar networks. *Annals of Operations Research*, 33(6):403–418, 1991.

[30] M. Bern and P. Plassmann. The Steiner problem with edge lengths 1 and 2. *Information Processing Letters*, 32:171–176, 1989.

[31] M. W. Bern. Faster exact algorithm for Steiner trees in planar networks. *Networks*, 20:109–120, 1990.

[32] D. Bienstock, E. F. Brickell, and C. L. Monma. On the structure of minimum-weight $k$-connected spanning networks. *SIAM Journal on Discrete Mathematics*, 3(3):320–329, 1990.

[33] K. D. Boese, A. B. Kahng, B. A. McCoy, and G. Robins. Rectilinear Steiner trees with minimum Elmore delay. In *Proceedings of the ACM Design Automation Conference (DAC)*, pages 381–386, San Francisco, California, 1994.

[34] K. D. Boese, A. B. Kahng, B. A. McCoy, and G. Robins. Near-optimal critical sink routing tree constructions. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 14(12):1417–1436, 1995.

[35] K. D. Boese, A. B. Kahng, and G. Robins. High-performance routing trees with identified critical sinks. In *Proceedings of the ACM Design Automation Conference (DAC)*, pages 182–187, Dallas, Texas, 1993.

[36] K. Bopp. *Üeber das kürzeste Verbindungssystem zwischen vier Punkten*. PhD thesis, Universität Göttingen, 1879.

[37] M. Borah, R. M. Owens, and M. J. Irwin. An edge-based heuristic for Steiner routing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13:1563–1568, 1994.

[38] W. M. Boyce. An improved program for the full Steiner tree problem. *ACM Transactions on Mathematical Software*, 3(4):359–385, 1977.

[39] W. M. Boyce and J. B. Seery. STEINER 72, An improved version of Cockayne and Schiller's program STEINER for the minimal network problem. Technical Report No. 35, Bell Laboratories, Murray Hill, NJ, 1973.

[40] S. P. Boyd, S.-J. Kim, D. D. Patil, and M. A. Horowitz. Digital circuit optimization via geometric programming. *Operations Research*, 53(6):899–932, 2005.

[41] E. Bozorgzadeh, R. Kastner, and M. Sarrafzadeh. Creating and exploiting flexibility in Steiner trees. In *Proceedings of the ACM Design Automation Conference (DAC'01)*, pages 195–198, Las Vegas, Nevada, 2001.

[42] E. Bozorgzadeh, R. Kastner, and M. Sarrafzadeh. Creating and exploiting flexibility in rectilinear Steiner trees. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 22(5):605–615, 2003.

[43] M. L. Brady, D. J. Brown, and K. Powers. Channel routing on a $60°$ grid. In *Proceedings of the Conference on Information Science and Systems*, pages 926–931, 1990.

[44] M. L. Brady, D. J. Brown, and K. Powers. Hexagonal models for channel routing. *Algorithmica*, 19:263–290, 1997.

[45] M. Brazil. Steiner minimum trees in uniform orientation metrics. In D.-Z. Du and X. Cheng, editors, *Steiner Trees in Industries*, pages 1–27. Kluwer Academic Publishers, Boston, 2001.

[46] M. Brazil, T. Cole, J. H. Rubinstein, D. A. Thomas, J. F. Weng, and N. C. Wormald. Minimal Steiner trees for $2^k \times 2^k$ square lattices. *Journal of Combinatorial Theory, Series A*, 73:91–110, 1996.

[47] M. Brazil, R. L. Graham, D. A. Thomas, and M. Zachariasen. On the history of the Euclidean Steiner tree problem. *Archive for History of Exact Sciences*, 68:327–354, 2014.

[48] M. Brazil, P. A. Grossman, D. H. Lee, J. H. Rubinstein, D. A. Thomas, and N. C. Wormald. Constrained path optimisation for underground mine layout. In *World Congress on Engineering 2007*, pages 856–861, London, UK, 2007.

[49] M. Brazil, D. H. Lee, J. H. Rubinstein, D. A. Thomas, J. F. Weng, and N. C. Wormald. Network optimisation of underground mine design. *Proceedings of the Australasian Institute of Mining and Metallurgy*, 305(1):57–66, 2000.

[50] M. Brazil, D. H. Lee, M. Van Leuven, J. H. Rubinstein, D. A. Thomas, and N. C. Wormald. Optimising declines in underground mines. *Mining Technology*, 112(3):164–170, 2003.

[51] M. Brazil, B. K. Nielsen, P. Winter, and M. Zachariasen. Rotationally optimal spanning and Steiner trees in uniform orientation metrics. *Computational Geometry: Theory and Applications*, 29:251–263, 2004.

[52] M. Brazil, C. J. Ras, K. J. Swanepoel, and D. A. Thomas. Generalised $k$-Steiner tree problems in normed planes. *Algorithmica*, In Press, 2014.

[53] M. Brazil, C. J. Ras, and D. A. Thomas. Approximating minimum Steiner point trees in Minkowski planes. *Networks*, 56(4):244–254, 2010.

[54] M. Brazil, C. J. Ras, and D. A. Thomas. The bottleneck 2-connected $k$-Steiner network problem for $k \leq 2$. *Discrete Applied Mathematics*, 160(7):1028–1038, 2012.

[55] M. Brazil, C. J. Ras, and D. A. Thomas. Relay augmentation for lifetime extension of wireless sensor networks. *IET Wireless Sensor Systems*, 3(2):145–152, 2013.

[56] M. Brazil, C. J. Ras, and D. A. Thomas. An exact algorithm for the bottleneck 2-connected $k$-Steiner network problem in $L_p$ planes. *arXiv preprint arXiv:1111.2105*, 2014.

[57] M. Brazil, C. J. Ras, and D. A. Thomas. A flow-dependent quadratic Steiner tree problem in the Euclidean plane. *Networks*, 64(1):18–28, 2014.

[58] M. Brazil, J. H. Rubinstein, D. A. Thomas, J. F. Weng, and N. C. Wormald. Full minimal Steiner trees on lattice sets. *Journal of Combinatorial Theory, Series A*, 78:51–91, 1997.

[59] M. Brazil, J. H. Rubinstein, D. A. Thomas, J. F. Weng, and N. C. Wormald. Minimal Steiner trees for rectangular arrays of lattice points. *Journal of Combinatorial Theory, Series A*, 79:181–208, 1997.

[60] M. Brazil, J. H. Rubinstein, D. A. Thomas, J. F. Weng, and N. C. Wormald. Gradient-constrained minimum networks. I. Fundamentals. *Journal of Global Optimization*, 21(2):139–155, 2001.

[61] M. Brazil, J. H. Rubinstein, D. A. Thomas, J. F. Weng, and N. C. Wormald. Gradient-constrained minimum networks. III. Fixed topology. *Journal of Optimization Theory and Applications*, 155(1):336–354, 2012.

[62] M. Brazil, D. A. Thomas, B. K. Nielsen, P. Winter, C. Wulff-Nilsen, and M. Zachariasen. A novel approach to phylogenetic trees: $d$-dimensional geometric Steiner trees. *Networks*, 53(2):104–111, 2009.

[63] M. Brazil, D. A. Thomas, and J. F. Weng. Gradient constrained minimal Steiner trees. In *Network Design: Connectivity and Facilities Location (DIMACS Series in Discrete Mathematics and Theoretical Computer Science, volume 40)*. American Mathematical Society, 1998.

[64] M. Brazil, D. A. Thomas, and J. F. Weng. A polynomial time algorithm for rectilinear Steiner trees with terminals constrained to curves. *Networks*, 33(2):145–155, 1999.

[65] M. Brazil, D. A. Thomas, and J. F. Weng. Minimum networks in uniform orientation metrics. *SIAM Journal on Computing*, 30:1579–1593, 2000.

[66] M. Brazil, D. A. Thomas, and J. F. Weng. On the complexity of the Steiner problem. *Journal of Combinatorial Optimization*, 4:187–195, 2000.

[67] M. Brazil, D. A. Thomas, and J. F. Weng. Rectilinear Steiner minimal trees on parallel lines. In D.-Z. Du, J. M. Smith, and J. H. Rubinstein, editors, *Advances in Steiner Trees*, pages 27–37. Kluwer Academic Publishers, Boston, 2000.

[68] M. Brazil, D. A. Thomas, and J. F. Weng. Upper and lower bounds for the lengths of Steiner trees in 3-space. *Geometriae Dedicata*, 109:107–119, 2004.

[69] M. Brazil, D. A. Thomas, and J. F. Weng. Gradient-constrained minimum networks (II). Labelled or locally minimal Steiner points. *Journal of Global Optimization*, 42(1):23–37, 2008.

[70] M. Brazil, D. A. Thomas, J. F. Weng, and M. Zachariasen. Canonical forms and algorithms for Steiner trees in uniform orientation metrics. *Algorithmica*, 44:281–300, 2006.

[71] M. Brazil and M. G. Volz. Gradient-constrained minimum interconnection networks. In P. M. Pardalos, D. Z. Du, and R. L. Graham, editors, *Handbook of Combinatorial Optimization*, pages 1459–1510. Springer, New York, 2013.

[72] M. Brazil, P. Winter, and M. Zachariasen. Flexibility of Steiner trees in uniform orientation metrics. In *Proceedings of the International Symposium on Algorithms and Computation (ISAAC), Lecture Notes in Computer Science 3341*, pages 196–208, Hong Kong, 2004.

[73] M. Brazil, P. Winter, and M. Zachariasen. Flexibility of Steiner trees in uniform orientation metrics. *Networks*, 46:142–153, 2005.

[74] M. Brazil and M. Zachariasen. Steiner trees for fixed orientation metrics. *Journal of Global Optimization*, 43:141–169, 2009.

[75] U. Brenner and J. Vygen. Worst-case ratios of networks in the rectilinear plane. *Networks*, 38:126–139, 2001.

[76] U. Brenner and J. Vygen. Analytical methods in VLSI placement. In C. J. Alpert, D. P. Mehta, and S. S. Sapatnekar, editors, *Handbook of Algorithms for VLSI Physical Design Automation*, chapter 17, pages 327–346. Taylor and Francis, Boca Raton, 2009.

[77] S. Burman, H. Chen, and N. Sherwani. Improved global routing using $\lambda$-geometry. In *Proceedings of the 29 Annual Allerton Conference on Communications, Computing and Controls*, Urbana, Illinois, 1991.

[78] L. L. Cavalli-Sforza and A. W. F. Edwards. Phylogenetic analysis: Models and estimation procedures. *Evolution*, 21:550–570, 1967.

[79] J. C. Cavendish. Automatic triangulation of arbitrary planar domains for the finite element method. *International Journal for Numerical Methods in Engineering*, 8(4):679–696, 1974.

[80] G. D. Chakerian and M. A. Ghandehari. The Fermat problem in Minkowski spaces. *Geometriae Dedicata*, 17:227–238, 1985.

[81] K. Chaudhary and P. Robinson. Channel routing by sorting. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 10:754–760, 1991.

[82] P. P. Chaudhuri. An ecological approach to wire routing. In *IEEE International Symposium on Circuits and Systems*, pages 854–857, 1979.

[83] C. Y. R. Chen, C. Y. Hou, and U. Singh. Optimal algorithms for bubble sort based non-Manhattan channel routing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13:603–609, 1994.

[84] D. Chen, D.-Z. Du, X.-D. Hu, G.-H. Lin, L. Wang, and G. Xue. Approximations for Steiner trees with minimum number of Steiner points. *Journal of Global Optimization*, 18(1):17–33, 2000.

[85] H. Chen, C. K. Cheng, A. B. Kahng, I. I. Mandoiu, and Q. Wang. Estimation of wirelength reduction for $\lambda$-geometry vs. Manhattan placement and routing. In *Proceedings ACM International Workshop on System Level Interconnect Prediction (SLIP)*, pages 71–76, Monterey, California, 2003.

[86] H. Chen, C. K. Cheng, A. B. Kahng, I. I. Mandoiu, Q. Wang, and B. Yao. The Y-architecture for on-chip interconnect: Analysis and methodology. In *Proceedings ACM International Conference on Computer-Aided Design (ICCAD)*, pages 13–19, 2003.

[87] H. Chen, C. K. Cheng, A. B. Kahng, I. I. Mandoiu, Q. Wang, and B. Yao. The Y-architecture for on-chip interconnect: Analysis and methodology. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24:588–599, 2005.

[88] H. Chen, B. Yao, F. Zhou, and C. K. Cheng. The Y-architecture: Yet another on-chip interconnect solution. In *Proceedings Asia and South Pacific Design Automation Conference*, pages 840–846, Kitakyushu, Japan, 2003.

[89] S.-W. Cheng. The Steiner tree problem for terminals on the boundary of a rectilinear polygon. *Theoretical Computer Science*, 237(1–2):213–238, 2000.

[90] S.-W. Cheng, A. Lim, and C.-T. Wu. Optimal rectilinear Steiner tree for extremal point sets. In K. W. Ng, P. Raghavan, N. V. Balasubramanian, and F. Y. L. Chin, editors, *Algorithms and Computation*, volume 762 of *Lecture Notes in Computer Science*, pages 523–532. Springer, Berlin-Heidelberg, 1993.

[91] S.-W. Cheng and C.-K. Tang. A fast algorithm for computing optimal rectilinear Steiner trees for extremal point sets. Technical Report HKUST-CS95-20, Department of Computer Science, HKUST, 1995.

[92] S.-W. Cheng and C.-K. Tang. A fast algorithm for computing optimal rectilinear Steiner trees for extremal point sets (Extended Abstact). In J. Staples, P. Eades, N. Katoh, and A. Moffat, editors, *Algorithms and Computations*, volume 1004 of *Lecture Notes in Computer Science*, pages 322–331. Springer, Berlin-Heidelberg, 1995.

[93] X. Cheng, D.-Z. Du, L. Wang, and B. Xu. Relay sensor placement in wireless sensor networks. *Wireless Networks*, 14(3):347–355, 2008.

[94] L. Paul Chew and Robert L. Drysdale III. Voronoi diagrams based on convex distance function. In *Proceedings of the First Annual ACM Symposium on Computational Geometry (SCG)*, pages 235–244, New York, NY, 1985.

[95] C. Chiang and M. Sarrafzadeh. Wirability of knock-knee layouts with 45-degree wires. *IEEE Transactions on Circuits and Systems*, 38:613–624, 1991.

[96] C. Chiang, M. Sarrafzadeh, and C. K. Wong. Global routing based on Steiner min-max trees. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 9:1318–1325, 1990.

[97] B. Choi, C. Chiang an J. Kawa, and M. Sarrafzadeh. Routing resources consumption on M-arch and X-arch. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, volume 5, pages V–73—V–76, Vancouver, British Columbia, Canada, 2004.

[98] S. Chopra, E. Gorres, and M. R. Rao. Solving the Steiner tree problem on a graph using branch and cut. *ORSA Journal on Computing*, 4:320–335, 1992.

[99] S. Chopra and M. R. Rao. The Steiner tree problem I: Formulations, compositions and extension of facets. *Mathematical Programming*, 64:209–229, 1994.

[100] S. Chopra and M. R. Rao. The Steiner tree problem II: Properties and classes of facets. *Mathematical Programming*, 64:231–246, 1994.

[101] S. A. Chowdhury, S. E. Shackney, K. Heselmeyer-Haddad, T. Ried, A. A. Schäffer, and R. Schwartz. Phylogenetic analysis of multiprobe fluorescence in situ hybridization data from tumor cell populations. *Bioinformatics*, 29:i189–i198, 2013.

[102] C. Chu. FLUTE: Fast lookup table based wirelength estimation technique. In *Proceedings ACM International Conference on Computer-Aided Design (ICCAD)*, pages 696–701, San Jose, California, 2004.

[103] C. Chu and Y.-C. Wong. FLUTE: Fast lookup table based rectilinear Steiner minimal tree algorithm for VLSI design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(1):70–83, 2008.

[104] F. Chung, M. Gardner, and R. L. Graham. Steiner trees on a checkerboard. *Mathematics Magazine*, 62(2):83–96, 1989.

[105] F. R. K. Chung and R. L. Graham. Steiner trees for ladders. *Annals of Discrete Mathematics*, 2:173–200, 1978.

[106] F. R. K. Chung and R. L. Graham. A new bound for Euclidean Steiner minimal trees. *Annals of the New York Academy of Sciences*, 440(1):328–346, 1985.

[107] D. Cieslik. The vertex degrees of Steiner minimal trees in Minkowski planes. In *Topics in Combinatorics and Graph Theory*, pages 201–206. Physica-Verlag, Heidelberg, 1990.

[108] D. Cieslik. *Shortest Connectivity — Introduction with Applications in Phylogeny*, volume 17 of *Combinatorial Optimization*. Springer, New York, 2004.

[109] D. Cieslik. The Steiner ratio of Banach-Minkowski spaces - a survey. In D. Z. Du and P. M. Pardalos, editors, *Handbook of Combinatorial Optimization, Supplement Volume B*, pages 55–81. Springer, New York, 2005.

[110] E. J. Cockayne. On the Steiner problem. *Canadian Mathematical Bulletin*, 10:431–450, 1967.

[111] E. J. Cockayne. On the efficiency of the algorithm for Steiner minimal trees. *SIAM Journal on Applied Mathematics*, 18(1):150–159, 1970.

[112] E. J. Cockayne and D. E. Hewgill. Exact computation of Steiner minimal trees in the plane. *Information Processing Letters*, 22:151–156, 1986.

[113] E. J. Cockayne and D. E. Hewgill. Improved computation of plane Steiner minimal trees. *Algorithmica*, 7(2/3):219–229, 1992.

[114] E. J. Cockayne and Z. A. Melzak. Euclidean constructibility in graph-minimization problems. *Mathematics Magazine*, 42(4):206–208, 1969.

[115] E. J. Cockayne and D. G. Schiller. Computation of Steiner minimal trees. In D. J. A. Welsh and D. R. Woodall, editors, *Combinatorics*, pages 52–71. Institute for Mathematics and Applications, Southend-on-Sea, Essex, England, 1972.

[116] C. J. Colbourn and G. Xue. Grade of service Steiner trees in series-parallel networks. In *Advances in Steiner Trees*, pages 163–174. Springer, 2000.

[117] T. Colthurst, C. Cox, J. Foisy, H. Howards, K. Kollett, H. Lowy, and S. Root. Networks minimizing length plus the number of Steiner points. In D.-Z. Du and P. M. Pardalos, editors, *Network Optimization Problems: Algorithms, Applications and Complexity*, pages 23–36. World Scientific, Singapore, 1993.

[118] J. Cong, L. He, C.-K. Koh, and P. H. Madden. Performance optimization of VLSI interconnect layout. *Integration, the VLSI Journal*, 21:1–94, 1996.

[119] J. Cong, A. B. Kahng, G. Robins, M. Sarrafzadeh, and C. K. Wong. Provably good performance-driven global routing. *Computer Aided Design*, 11(6):739–752, 1992.

[120] J. Cong, K. S. Leung, and D. Zhou. Performance-driven interconnect design based on distributed RC delay model. In *Proceedings of the ACM Design Automation Conference (DAC)*, pages 606–611, Dallas, Texas, 1993.

[121] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, Cambridge, 2001.

[122] C. S. Coulston. Constructing exact octagonal Steiner minimal trees. In *Proceedings of the 13th ACM Great Lakes Symposium on VLSI (GLSVLSI)*, pages 1–6, Washington, DC, 2003.

[123] R. Courant and H. Robbins. *What Is Mathematics?* Oxford University Press, London, 1941.

[124] C. L. Cox. Flow-dependent networks: Existence and behavior at Steiner points. *Networks*, 31(3):149–156, 1998.

[125] H. S. M. Coxeter. *Introduction to Geometry*. John Wiley & Sons Inc., New York, 1969.

[126] J. R. Current, C. S. ReVelle, and J. L. Cohon. The hierarchical network design problem. *European Journal of Operational Research*, 27(1):57–66, 1986.

[127] S. Vahdati Daneshmand. *Algorithmic Approaches to the Steiner Problem in Networks*. PhD thesis, Universität Mannheim, 2004.

[128] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry. Algorithms and Applications*. Springer-Verlag, Berlin, Germany, 3rd edition, 2008.

[129] P. O. de Wet. *Geometric Steiner Minimal Trees*. PhD thesis, UNISA, Pretoria, 2009.

[130] R. F. DeMar. The problem of the shortest network joining $n$ points. *Mathematics Magazine*, 41(5):225–231, 1968.

[131] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[132] S. E. Dreyfus and R. A. Wagner. The Steiner problem in graphs. *Networks*, 1:195–207, 1972.

[133] Z. Drezner and G. O. Wesolowsky. A new method for the multifacility minimax location problem. *Journal of the Operational Research Society*, 29(11):1095–1101, 1978.

[134] D.-Z. Du, B. Gao, R. L. Graham, Z.-C. Liu, and P.-J. Wan. Minimum Steiner trees in normed planes. *Discrete and Computational Geometry*, 9:351–370, 1993.

[135] D.-Z. Du and X.-D. Hu. *Steiner Tree Problems in Computer Communication Networks*. World Scientific Publishing Company, Singapore, 2008.

[136] D.-Z. Du and F. K. Hwang. A proof of Gilbert and Pollak's conjecture on the Steiner ratio. *Algorithmica*, 7:121–135, 1992.

[137] D.-Z. Du and F. K. Hwang. Reducing the Steiner problem in a normed space. *SIAM Journal on Computing*, 21:1001–1007, 1992.

[138] D.-Z. Du, F. K. Hwang, G. D. Song, and G. Y. Ting. Steiner minimal trees on sets of four points. *Discrete and Computational Geometry*, 2:401–414, 1987.

[139] D.-Z. Du, F. K. Hwang, and J. F. Weng. Steiner minimal trees for regular polygons. *Discrete and Computational Geometry*, 2:65–84, 1987.

[140] D.-Z. Du, L. Wang, and B. Xu. The Euclidean bottleneck Steiner tree and Steiner tree with minimum number of Steiner points. In *Computing and Combinatorics*, pages 509–518. Springer, 2001.

[141] C. W. Duin. *Steiner's Problem in Graphs - Approximation, Reduction, Variation*. PhD thesis, University of Amsterdam, Netherlands, 1993.

[142] C. W. Duin. Preprocessing the Steiner problem in graphs. In D.-Z. Du, J. M. Smith, and J. H. Rubinstein, editors, *Advances in Steiner Trees*, pages 173–233. Kluwer Academic Publishers, Boston, 2000.

[143] C. W. Duin and A. Volgenant. An edge elimination test for the Steiner problem in graphs. *Operations Research Letters*, 8(2):79–83, 1989.

[144] C. W. Duin and A. Volgenant. Reduction tests for the Steiner problem in graphs. *Networks*, 19(5):549–567, 1989.

[145] M. Durand, J.-F. Sadoc, and D. Weaire. Maximum electrical conductivity of a network of uniform wires: The Lemlich law as an upper bound. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 460(2045):1269–1284, 2004.

[146] R. Durier and C. Michelot. Geometrical properties of the Fermat-Weber problem. *European Journal of Operational Research*, 20:322–343, 1985.

[147] P. Dutta, S. P. Khastgir, and A. Roy. Steiner trees and spanning trees in six-pin soap films. *American Journal of Physics*, 78:215–221, 2010.

[148] J. Edmonds. Submodular functions, matroids, and certain polyhedra. *Combinatorial Structures and Their Applications*, pages 69–87, 1970.

[149] W. C. Elmore. The transient response of damped linear networks with particular regard to wideband amplifiers. *Journal of Applied Physics*, 19:55–63, 1948.

[150] J. Elzinga, D. Hearn, and W. D. Randolph. Minimax multifacility location with Euclidean distances. *Transportation Science*, 10(4):321–336, 1976.

[151] M. Fampa and K. M. Anstreicher. An improved algorithm for computing Steiner minimal trees in Euclidean $d$-space. *Discrete Optimization*, 5:530–540, 2008.

[152] J. S. Farris. Methods for computing Wagner trees. *Systematic Zoology*, 19(1):83–92, 1970.

[153] P. Fermat. *Oeuvres*, volume 1. Gauthier-Villars, Paris, 1891.

[154] U. Fößmeier and M. Kaufmann. On exact solutions for the rectilinear Steiner tree problem. Technical Report WSI-96-09, Universität Tübingen, 1996.

[155] U. Fößmeier and M. Kaufmann. Solving rectilinear Steiner tree problems exactly in theory and practice. In R. Burkard and G. Woeginger, editors, *Algorithms ESA 97*, volume 1284 of *Lecture Notes in Computer Science*, pages 171–185. Springer, Berlin-Heidelberg, 1997.

[156] U. Fößmeier and M. Kaufmann. On exact solutions for the rectilinear Steiner tree problem. Part I: Theoretical results. *Algorithmica*, 26:68–99, 2000.

[157] L. R. Foulds and R. L. Graham. The Steiner problem in phylogeny is NP-complete. *Advances in Applied Mathematics*, 3:43–49, 1982.

[158] I. Frommer, B. Golden, and G. Pundoor. Heuristic methods for solving Euclidean non-uniform Steiner tree problems. In *The Next Wave in Computing, Optimization, and Decision Technologies*, volume 29 of *Operations Research/Computer Science Interfaces*, pages 133–148. Springer, New York, 2005.

[159] B. Fuchs, W. Kern, D. Molle, S. Richter, P. Rossmanith, and X. Wang. Dynamic programming for minimum Steiner trees. *Theory of Computing Systems*, 41(3):493–500, 2007.

[160] B. Fuchs, W. Kern, and X. Wang. The number of tree stars is $O^*(1.357^k)$. *Electronic Notes in Discrete Mathematics*, 25:183–185, 2006.

[161] J. L. Ganley. *Geometric Interconnection and Placement Algorithms*. PhD thesis, The University of Virginia, USA, 1995.

[162] J. L. Ganley and J. P. Cohoon. A faster dynamic programming algorithm for exact rectilinear Steiner minimal trees. In *Proceedings of the Fourth Great Lakes Symposium on VLSI*, pages 238–241, South Bend, Indiana, 1994.

[163] J. L. Ganley and J. P. Cohoon. Optimal rectilinear Steiner minimal trees in $o(n^2 2.62^n)$ time. In *Proceedings of the Sixth Canadian Conference on Computational Geometry*, pages 308–313, Saskatoon, Saskatchewan, 1994.

[164] J. L. Ganley and J. P. Cohoon. Routing a multi-terminal critical net: Steiner tree construction in the presence of obstacles. In *IEEE Proceedings of the International Symposium on Circuits and Systems*, pages 113–116, London, 1994.

[165] J. L. Ganley and J. P. Cohoon. Improved computation of optimal rectilinear Steiner minimal trees. *International Journal of Computational Geometry and Applications*, 7(5):457–472, 1997.

[166] J. L. Ganley and J. S. Salowe. Optimal and approximate bottleneck Steiner trees. *Operations Research Letters*, 19:217–224, 1996.

[167] J. L. Ganley and J. S. Salowe. The power-$p$ Steiner tree problem. *Nordic Journal of Computing*, 5(2):115–127, 1998.

[168] B. Gao, D.-Z. Du, and R. L. Graham. The tight lower bound for the Steiner ratio in Minkowski planes. In *Proceedings of the Tenth Annual Symposium on Computational Geometry*, pages 183–191, Stony Brook, New York, 1994.

[169] M. R. Garey, R. L. Graham, and D. S. Johnson. The complexity of computing Steiner minimal trees. *SIAM Journal on Applied Mathematics*, 32(4):835–859, 1977.

[170] M. R. Garey and D. S. Johnson. The rectilinear Steiner tree problem is NP-complete. *SIAM Journal on Applied Mathematics*, 32(4):826–834, 1977.

[171] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman, San Francisco, CA, 1979.

[172] C. F. Gauss and H. C. Schumacher. *Briefwechsel Zwischen C. F. Gauss und H. C. Schumacher*. G. Esch, 1861.

[173] G. Georgakopoulos and C. H. Papadimitriou. The 1-Steiner tree problem. *Journal of Algorithms*, 8:122–130, 1987.

[174] S. H. Gerez. *Algorithms for VLSI Design Automation*. John Wiley & Sons, Inc., New York, 1999.

[175] J. D. Gergonne. Solutions purement géométriques des problèmes de minimis proposés aux pages 196, 232 et 292 de ce volume, et de divers autres problèmes analogues. *Annales de Mathématiques pures et appliquées*, 1:375–384, 1811.

[176] M. Gester, D. Müller, T. Nieberg, C. Panten, C. Schulte, and J. Vygen. Algorithms and data structures for fast and good VLSI routing. In *ACM Proceedings of the 49th Annual Design Automation Conference (DAC)*, pages 459–464, New York, NY, 2012.

[177] M. Ghandehari and E. J. O'Neill. The reflection property in normed linear planes with applications to generalized conics. Technical Report TR - 351, University of Texas Arlington, 2005.

[178] S. K. Ghosh and D. M. Mount. An output-sensitive algorithm for computing visibility graphs. *SIAM Journal on Computing*, 20(5):888–910, 1991.

[179] E. N. Gilbert. Minimum cost communication networks. *Bell System Technical Journal*, 46:2209–2227, 1967.

[180] E. N. Gilbert and H. O. Pollak. Steiner minimal trees. *SIAM Journal on Applied Mathematics*, 16(1):1–29, 1968.

[181] M. X. Goemans. The Steiner tree polytope and related polyhedra. *Mathematical Programming*, 63:157–182, 1994.

[182] M. X. Goemans and D. J. Bertsimas. Survivable networks, linear programming relaxations and the parsimonious property. *Mathematical Programming*, 60:145–166, 1993.

[183] M. X. Goemans and Y. S. Myung. A catalog of Steiner tree formulations. *Networks*, 23:19–28, 1993.

[184] R. L. Graham. Some results on Steiner minimal trees. Unpublished manuscript, 11 May 1967.

[185] R. L. Graham and F.K. Hwang. A remark on Steiner minimal trees. *Bulletin of the Institute of Mathematics Academia Sinica*, 4(1):177–182, 1976.

[186] S. Gueron and R. Tessler. The Fermat-Steiner problem. *The American Mathematical Monthly*, 109(5):443–451, 2002.

[187] S. L. Hakimi. Steiner's problem in graphs and its implications. *Networks*, 1:113–133, 1971.

[188] H. Hamrin. Underground mining methods and applications. In W. A. Hustrilid and R. L. Bullock, editors, *Underground Mining Methods*. Society for Mining, Metallurgy, and Exploration, 2001.

[189] M. Hanan. On Steiner's problem with rectilinear distance. *SIAM Journal on Applied Mathematics*, 14(2):255–265, 1966.

[190] F. C. Harris. Steiner minimal trees: An introduction, parallel computation, and future work. In D. Z. Du and P. M. Pardalos, editors, *Handbook of Combinatorial Optimization*, volume 2, pages 105–157. Kluwer Academic Publishers, New York, 1998.

[191] M. Hayase. Exact location of the Steiner point in the three-point Steiner minimum tree for $\lambda$-geometry. *Electronics and Communications in Japan*, 84:84–94, 2001.

[192] F. Heinen. *Über Systeme von Kräften*. G. D. Bädeker, 1834.

[193] S. Heiss. A path connection algorithm for multi-layer boards. In *ACM Proceedings of the 5th Annual Design Automation Workshop (DAC)*, pages 6.1–6.14, New York, NY, 1968.

[194] S. Held, B. Korte, D. Rautenbach, and J. Vygen. Combinatorial optimization in VLSI design. In V. Chvatal, editor, *Combinatorial Optimization: Methods and Applications*, pages 33–96. IOS Press, Amsterdam, 2011.

[195] S. Held and D. Rotter. Shallow-light Steiner arborescences with vertex delays. In M. Goemans and J. Correa, editors, *Integer Programming and Combinatorial Optimization*, volume 7801 of *Lecture Notes in Computer Science*, pages 229–241. Springer, Berlin-Heidelberg, 2013.

[196] S. Held and S. T. Spirkl. A fast algorithm for rectilinear Steiner trees with length restrictions on obstacles. In *Proceedings of the 2014 ACM International Symposium on Physical Design (ISPD)*, pages 37–44, Petaluma, California, 2014.

[197] A. Hetzel. *Verdrahtung im VLSI-Design: Spezielle Teilprobleme und ein sequentielles Lösungsverfahren*. PhD thesis, Research Institute for Discrete Mathematics, University of Bonn, 1995.

[198] D. Hightower. The interconnection problem: A tutorial. *Computer*, 7:18–32, 1974.

[199] J.-M. Ho, G. Vijayan, and C. K. Wong. New algorithms for the rectilinear Steiner tree problem. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 9(2):185–193, 1990.

[200] T.-Y. Ho, C.-F. Chang, Y.-W. Chang, and S.-J. Chen. Multilevel full-chip routing for the X-based architecture. In *Proceedings of the ACM Design Automation Conference (DAC)*, pages 597–602, Anaheim, California, 2005.

[201] E. Hoffmann. Über das kürzeste Verbindungssystem zwischen vier Punkten der Ebene. In *Program des Königlichen Gymnasiums zu Wetzlar für das Schuljahr von Ostern 1889 bis Ostern 1890*. Schnitzler, 1890.

[202] J. E. Hofmann. Elementare Lösung einer Minimumsaufgabe. *Zeitschrift für mathematischen und naturwissenschaftligen Unterricht*, 60:22–23, 1929.

[203] R. Honsberger. *Mathematical Gems*. Mathematical Association of America, Washington, D. C., 1973.

[204] H. Hou, J. Hu, and S. S. Sapatnekar. Non-Hanan routing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 18(4):436–444, 1999.

[205] S. Hougardy, J. Silvanus, and J. Vygen. Dijkstra meets Steiner: A fast exact goal-oriented Steiner tree algorithm. Technical report, Research Institute for Discrete Mathematics, University of Bonn, 2014.

[206] T. Huang, L. Li, and E. F. Y. Young. On the construction of optimal obstacle-avoiding rectilinear Steiner minimum trees. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 30(5):718–731, 2011.

[207] T. Huang and E. F. Y. Young. Obstacle-avoiding rectilinear Steiner minimum tree construction: An optimal approach. In *Proceedings ACM International Conference on Computer-Aided Design (ICCAD)*, pages 610–613, San Jose, California, 2010.

[208] T. Huang and E F. Y. Young. An exact algorithm for the construction of rectilinear Steiner minimum trees among complex obstacles. In *Proceedings of the 48th ACM/IEEE Design Automation Conference (DAC)*, pages 164–169, San Diego, California, 2011.

[209] T. Huang and E. F. Y. Young. Construction of rectilinear Steiner minimum trees with slew constraints over obstacles. In *Proceedings ACM International Conference on Computer-Aided Design (ICCAD)*, pages 144–151, New York, NY, 2012.

[210] T. Huang and E. F. Y. Young. Obsteiner: An exact algorithm for the construction of rectilinear Steiner minimum trees in the presence of complex rectilinear obstacles. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(6):882–893, 2013.

[211] F. K. Hwang. On Steiner minimal trees with rectilinear distance. *SIAM Journal on Applied Mathematics*, 30:104–114, 1976.

[212] F. K. Hwang. A linear time algorithm for full Steiner trees. *Operations Research Letters*, 4(5):235–237, 1986.

[213] F. K. Hwang, D. S. Richards, and P. Winter. *The Steiner Tree Problem*. Annals of Discrete Mathematics 53. Elsevier Science Publishers, Netherlands, 1992.

[214] F. K. Hwang and J. F. Weng. The shortest network under a given topology. *Journal of Algorithms*, 13:468–488, 1992.

[215] F. K. Hwang, J. F. Weng, and D. Z. Du. A class of full Steiner minimal trees. *Discrete Mathematics*, 45(1):107–112, 1983.

[216] M. Igarashi, T. Mitsuhashi, A. Le, S. Kazi, Y.-T. Lin, A. Fujimura, and S. Teig. A diagonal interconnect architecture and its application to RISC core design. In *IEEE Proceedings of the International Solid-State Conference*, pages 460–461, 2002.

[217] E. Ihler. The rectilinear class Steiner tree problem for intervals on two parallel lines. *Mathematical Programming*, 63(3):281–296, 1994.

[218] E. Ihler, G. Reich, and P. Widmayer. Class Steiner trees and VLSI-design. *Discrete Applied Mathematics*, 90:173–194, 1999.

[219] D. P. Il'yutko. Locally minimal trees in $n$-normed spaces. *Mathematical Notes*, 74:619–629, 2003.

[220] N. Innami, B. H. Kim, Y. Mashiko, and K. Shiohama. The Steiner ratio conjecture of Gilbert-Pollak may still be open. *Algorithmica*, 57(4):869–872, 2010.

[221] A. O. Ivanov and A. A. Tuzhilin. Immersed polygons and their diagonal triangulations. *Izvestiya: Mathematics*, 72(1):63, 2008.

[222] A. O. Ivanov and A. A. Tuzhilin. The Steiner ratio Gilbert-Pollak conjecture is still open. *Algorithmica*, 62(1-2):630–632, 2012.

[223] A. O. Ivanov and A. A. Tuzhilin. Du-Hwang characteristic area: Catch-22. *ArXiv e-prints*, 2014.

[224] G. Jalal and J. Krarup. Geometrical solution to the Fermat problem with arbitrary weights. *Annals of Operations Research*, 123:67–104, 2003.

[225] V. Jarník and M. Kössler. O minimálních grafeth obeahujících n daných bodú. *Cas. Pest. Mat. a Fys.*, 63:223–235, 1934.

[226] T. Jiang, Z. Miller, and D. Pritkin. Near optimal bounds for Steiner trees in the hypercube. *SIAM Journal on Computing*, 40:1340–1360, 2011.

[227] D. D. Juhl. Full Steiner trees for the obstacle-avoiding rectilinear Steiner tree problem. Master's thesis, Department of Computer Science, University of Copenhagen, 2013.

[228] A. B. Kahng, J. Lienig, I. L. Markov, and J. Hu. *VLSI Physical Design: From Graph Partitioning to Timing Closure*. Springer Science+Business Media, 2011.

[229] A. B. Kahng, I. I. Mandoiu, and A. Z. Zelikovsky. Highly scalable algorithms for rectilinear and octilinear Steiner trees. In *Proceedings of the Asia and South Pacific Design Automation Conference*, pages 827–833, New York, 2003.

[230] A. B. Kahng and G. Robins. A new class of iterative Steiner tree heuristics with good performance. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11(7):893–902, 1992.

[231] A. B. Kahng and G. Robins. *On Optimal Interconnections for VLSI*. Kluwer Academic Publishers, Boston, 1995.

[232] H. Karl and A. Willig. *Protocols and Architectures for Wireless Sensor Networks*. John Wiley & Sons, New York, 2007.

[233] R. M. Karp. Reducibility among combinatorial problems. In J. W. Thatcher, editor, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972.

[234] S. Khuller, B. Raghavachari, and N. Young. Balancing minimum spanning trees and shortest-path trees. *Algorithmica*, 14:305–321, 1995.

[235] D. Kirszenblat. The Steiner ratio conjecture for eight points. Master's thesis, The University of Melbourne, 2014.

[236] T. Koch and A. Martin. Solving Steiner tree problems in graphs to optimality. *Networks*, 33:207–232, 1998.

[237] C.-K. Koh. Steiner problem in octilinear routing model. Master's thesis, National University of Singapore, 1995.

[238] C.-K. Koh and P. H. Madden. Manhattan or non-Manhattan? A study of alternative VLSI routing architectures. In *Proceedings of the 10th ACM Great Lakes Symposium on VLSI (GLSVLSI)*, pages 47–52, Evanston, Illinois, 2000.

[239] B. Korte and J. Vygen. *Combinatorial Optimization: Theory and Algorithms*. Algorithms and Combinatorics. Springer, 4th edition, 2008.

[240] B. Korte and J. Vygen. Combinatorial problems in chip design. In M. Grötschel and G. O. H. Katona, editors, *Building Bridges Between Mathematics and Computer Science*, pages 333–368. Springer, Berlin, 2008.

[241] J. Krarup and S. Vajda. On Torricelli's geometrical solution to a problem of Fermat. *IMA Journal of Management Mathematics*, 8(3):215–224, 1997.

[242] Y. S. Kupitz and H. Martini. Geometric aspects of the generalized Fermat-Torricelli problem. In I. Barany and K. Boroczky, editors, *Bolyai Society Mathematical Studies, 6: Intuitive Geometry*, pages 55–127. Janos Bolyai Mathematical Society, Budapest, 1997.

[243] G. Lawlor and F. Morgan. Paired calibrations applied to soap films, immiscible fluids, and surfaces or networks minimizing other norms. *Pacific Journal of Mathematics*, 166:55–83, 1994.

[244] C. Y. Lee. An algorithm for path connections and its applications. *IRE Transactions on Electronic Computers*, EC-10:346–365, 1961.

[245] D. T. Lee, T. H. Chen, and C. D. Yang. Shortest rectilinear paths among weighted obstacles. In *Proceedings of the Sixth Annual ACM Symposium on Computational Geometry (SCG)*, pages 301–310, New York, NY, 1990.

[246] D. T. Lee and C. F. Shen. The Steiner minimal tree problem in the $\lambda$-geometry plane. In *Proceedings of the International Symposium on Algorithms and Computation (ISAAC), Lecture Notes in Computer Science 1178*, pages 247–255, 1996.

[247] D. T. Lee, C. F. Shen, and C. L. Ding. On Steiner tree problem with 45 degree routing. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1680–1683, Seattle, Washington, 1995.

[248] D. T. Lee, C. D. Yang, and T. H. Chen. Shortest rectilinear paths among weighted obstacles. *International Journal of Computational Geometry and Applications*, 1:109–124, 1991.

[249] J. Lee. *A first course in combinatorial optimization*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 2004.

[250] K. K. Lee and H. W. Leong. SOAR: A channel router for octilinear routing model. In *Proceedings of the IEEE Asia-Pacific Conference on Circuits and Systems*, pages 346–351, Sydney, Australia, 1992.

[251] T. Lengauer. *Combinatorial Algorithms for Integrated Circuit Layout*. John Wiley & Sons, Chichester, England, 1990.

[252] H. Lerchs and I. F. Grossmann. Optimum design of open-pit mines. *Trans. Canad. Inst. Mining, Metallurgy, Petroleum*, 68:17–24, 1965.

[253] A. Y. Levin. Algorithm for the shortest connection of a group of graph vertices. *Soviet Mathematics Doklady*, 12:1477–1481, 1971.

[254] A. Levy. Energy-minimizing networks meet only in threes. *Journal of Undergraduate Mathematics*, 22:53–59, 1990.

[255] C.-S. Li, F. F.-K. Tong, C. J. Georgiou, and M. Chen. Gain equalization in metropolitan and wide area optical networks using optical amplifiers. In *Proceedings of the 13th IEEE Conference on Computer Communications: Networking for Global Communications (INFOCOM)*, pages 130–137, Toronto, Ontario, Canada, 1994.

[256] L. Li and E. F. Young. Obstacle-avoiding rectilinear Steiner tree construction. In *Proceedings ACM International Conference on Computer-Aided Design (ICCAD)*, pages 523–528, San Jose, California, 2008.

[257] Y. Y. Li, S. K. Cheung, K. S. Leung, and C. K. Wong. Steiner tree constructions in $\lambda_3$-metric. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 45(5):563–574, 1998.

[258] Y. Y. Li, K. S. Leung, and C. K. Wong. Efficient heuristics for orientation metric and Euclidean Steiner tree problems. *Journal of Combinatorial Optimization*, 4:79–98, 2000.

[259] Y. Y. Li, K. S. Leung, and C. K. Wong. Steiner trees in general nonuniform orientations. *Computing*, 66:41–78, 2001.

[260] C.-W. Lin, S.-Y. Chen, C.-F. Li, Y.-W. Chang, and C.-L. Yang. Obstacle-avoiding rectilinear steiner tree construction based on spanning graphs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(4):643–653, 2008.

[261] G.-H. Lin and G. Xue. Reducing the Steiner problem in an $A_3$-geometry plane. Manuscript, 1998.

[262] G.-H. Lin and G. Xue. The Steiner tree problem in $\lambda_4$-geometry plane. In *Proceedings of the International Symposium on Algorithms and Computation (ISAAC), Lecture Notes in Computer Science 1533*, pages 327–337, 1998.

[263] G.-H. Lin and G. Xue. Steiner tree problem with minimum number of Steiner points and bounded edge-length. *Information Processing Letters*, 69(2):53–57, 1999.

[264] G.-H. Lin and G. Xue. Reducing the Steiner problem in four uniform orientations. *Networks*, 35:287–301, 2000.

[265] C.-H. Liu, S.-Y. Kuo, D. T. Lee, C.-S. Lin, J.-H. Weng, and S.-Y. Yuan. Obstacle-avoiding rectilinear Steiner tree construction: A Steiner-point-based algorithm. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 31(7):1050–1060, 2012.

[266] E. Lodi. Routing multiterminal nets in a diagonal model. In *Proceedings of the 1988 Conference on Information Sciences and Systems*, pages 899–902, Princeton, New Jersey, 1988.

[267] E. Lodi, F. Luccio, and L. Pagli. A preliminary study of a diagonal channel-routing model. *Algorithmica*, 4:585–597, 1989.

[268] E. Lodi, F. Luccio, and L. Pagli. Routing in times square mode. *Information Processing Letters*, 35:41–48, 1990.

[269] E. Lodi, F. Luccio, and X. Song. A 2D channel router for the diagonal model. *Integration, the VLSI Journal*, 11:111–125, 1991.

[270] J. Long, H. Zhou, and S. O. Memik. EBOARST: An efficient edge-based obstacle-avoiding rectilinear Steiner tree construction algorithm. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(12):2169–2182, 2008.

[271] L. Lovász. *The matroid matching problem*. Algebraic methods in graph theory. Colloquia Mathematica Societatis János Bolyai, Szeged, Hungary, 1978.

[272] B. Lu, J. Gu, X. Hu, and E. Shragowitz. Wire segmenting for buffer insertion based on RSTP-MSP. *Theoretical Computer Science*, 262(1):257–267, 2001.

[273] T. L. Magnanti and L. A. Wolsey. Optimal trees. In C. L. Monma M. O. Ball, T. L. Magnanti and G. L. Nemhauser, editors, *Network Models*, volume 7 of *Handbooks in Operations Research and Management Science*, pages 503 – 615. Elsevier, 1995.

[274] I. I. Măndoiu, V. V. Vazirani, and J. L. Ganley. A new heuristic for rectilinear Steiner trees. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 19:1129–1139, 2000.

[275] I. I. Măndoiu and A. Z. Zelikovsky. A note on the MST heuristic for bounded edge-length Steiner trees with minimum number of Steiner points. *Information Processing Letters*, 75(4):165–167, 2000.

[276] H. Martini, K. J. Swanepoel, and G. Weiß. The geometry of Minkowski spaces – A survey. Part I. *Expositiones Mathematicae*, 19(2):97–142, 2001.

[277] H. Martini, K. J. Swanepoel, and G. Weiß. The Fermat-Torricelli problem in normed planes and spaces. *Journal of Optimization Theory and Applications*, 115:283–314, 2002.

[278] J. Maßberg and T. Nieberg. Rectilinear paths with minimum segment lengths. *Discrete Applied Mathematics*, 161(12):1769–1775, 2013.

[279] S. Mehlhos. Simple counter examples for the unsolvability of the Fermat- and Steiner-Weber-problem by compass and ruler. *Beiträge zur Algebra und Geometrie*, 41(1):151–158, 2000.

[280] Z. A. Melzak. On the problem of Steiner. *Canadian Mathematical Bulletin*, 4(2):143–148, 1961.

[281] Z. A. Melzak. *Companion to Concrete Mathematics*, volume II. John Wiley & Sons, New York, 1976.

[282] W. Miehle. Link-length minimization in networks. *Operations Research*, 6(2):232–243, 1958.

[283] Z. Miller and M. Perkel. The Steiner tree problem in the hypercube. *Networks*, 22:1–19, 1992.

[284] Z. Miller and D. Pritkin. Applying a result of Frankl and Rödl to the construction of Steiner trees in the hypercube. *Discrete Mathematics*, 131:183–194, 1994.

[285] D. Moffitt. Global routing revisited. In *Proceedings ACM International Conference on Computer-Aided Design (ICCAD)*, pages 805–808, New York, NY, 2009.

[286] M. D. Moffitt, J. A. Roy, and I. L. Markov. The coming of age of (academic) global routing. In *Proceedings of the 2008 ACM International Symposium on Physical Design (ISPD)*, pages 148–155, New York, NY, 2008.

[287] R. Möhring, D. Wagner, and F. Wagner. VLSI network design. In *Handbooks in Operations Research and Management Science*, volume 7, Amsterdam, 1995. Elsevier Science.

[288] C. Monma and S. Suri. Transitions in geometric minimum spanning trees. *Discrete & Computational Geometry*, 8:265–293, 1992.

[289] D. Müller. *Fast Resource Sharing in VLSI Routing*. PhD thesis, Research Institute for Discrete Mathematics, University of Bonn, 2009.

[290] M. Müller-Hannemann and S. Peyer. Approximation of rectilinear Steiner trees with length restrictions on obstacles. In *Workshop on Algorithms and Data Structures (WADS), Lecture Notes in Computer Science 2748*, pages 207–218, 2003.

[291] M. Müller-Hannemann and A. Schulze. Hardness and approximation of octilinear Steiner trees. *International Journal of Computational Geometry and Applications*, 17:231–260, 2007.

[292] J. Naor and B. Schieber. Improved approximations for shallow-light spanning trees. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science,* pages 536–541, Miami Beach, Florida, 1997.

[293] G. Narasimhan and M. Smid. *Geometric Spanner Networks*. Cambridge University Press, Cambridge, 2007.

[294] S. Natarajan, N. Sherwani, N. D. Holmes, and M. Sarrafzadeh. Over-the-cell channel routing for high performance circuits. In *Proceedings of the 29th ACM/IEEE Design Automation Conference (DAC),* pages 600–603, Los Alamitos, California, 1992.

[295] A. M. Newman, E. Rubio, R. Caro, A. Weintraub, and K. Eurek. A review of operations research in mine planning. *Interfaces*, 40(3):222–245, 2010.

[296] T. Nieberg. Gridless pin access in detailed routing. In *Proceedings of the 48th ACM/IEEE Design Automation Conference (DAC),* pages 170 –175, San Diego, California, 2011.

[297] B. K. Nielsen, P. Winter, and M. Zachariasen. An exact algorithm for the uniformly-oriented Steiner tree problem. In *Proceedings of the 10th European Symposium on Algorithms, Lecture Notes in Computer Science,* volume 2461, pages 760–772, Rome, Italy, 2002. Springer.

[298] B. K. Nielsen, P. Winter, and M. Zachariasen. Rectilinear trees under rotation and related problems. In *Proceedings of the 18th European Workshop on Computational Geometry,* pages 18–22, Warsaw, Poland, 2002.

[299] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. John Wiley & Sons, New York, 2nd edition, 1995.

[300] F. J. Oppermann, C. A. Boano, and K. Römer. A decade of wireless sensing applications: Survey and taxonomy. In H. M. Ammari, editor, *The Art of Wireless Sensor Networks,* pages 11–50. Springer, 2014.

[301] M. H. Pagh. Steiner trees in weighted fixed orientation metrics. Master's thesis, Department of Computer Science, University of Copenhagen, 2005.

[302] M. Paluszewski. VLSI routing in uniform orientation metrics. Master's thesis, Department of Computer Science, University of Copenhagen, 2004.

[303] M. Paluszewski, P. Winter, and M. Zachariasen. A new paradigm for general architecture routing. In *Proceedings of the 14th ACM Great Lakes Symposium on VLSI (GLSVLSI)*, pages 202–207, Boston, Massachusetts, 2004.

[304] M. Pecht and Y. T. Wong. *Advanced Routing of Electronic Modules*. CRC Press, New York, 1996.

[305] S. Peyer. Elmore-delay-optimale Steinerbäume im VLSI-Design. Master's thesis, Research Institute for Discrete Mathematics, University of Bonn, 2000.

[306] S. Peyer. *Shortest Paths and Steiner Trees in VLSI Routing*. PhD thesis, Research Institute for Discrete Mathematics, University of Bonn, 2007.

[307] S. Peyer, M. Zachariasen, and D. G. Jørgensen. Delay-related secondary objectives for rectilinear Steiner minimum trees. *Discrete Applied Mathematics*, 136:271–298, 2004.

[308] H. O. Pollak. Some remarks on the Steiner problem. *Journal of Combinatorial Theory, Series A*, 24(3):278–295, 1978.

[309] T. Polzin. *Algorithms for the Steiner Problem in Networks*. PhD thesis, Universität des Saarlandes, 2003.

[310] T. Polzin and S. Vahdati Daneshmand. A comparison of Steiner tree relaxations. *Discrete Applied Mathematics*, 112:241–261, 2001.

[311] T. Polzin and S. Vahdati Daneshmand. Improved algorithms for the Steiner problem in networks. *Discrete Applied Mathematics*, 112:263–300, 2001.

[312] T. Polzin and S. Vahdati Daneshmand. Extending reduction techniques for the Steiner tree problem. In R. Möhring and R. Raman, editors, *Algorithms ESA 2002*, volume 2461 of *Lecture Notes in Computer Science*, pages 795–807. Springer, Berlin-Heidelberg, 2002.

[313] T. Polzin and S. Vahdati Daneshmand. On Steiner trees and minimum spanning trees in hypergraphs. *Operations Research Letters*, 31:12–20, 2003.

[314] T. Polzin and S. Vahdati Daneshmand. Approaches to the Steiner problem in networks. *Lecture Notes in Computer Science*, 5515:81–103, 2009.

[315] K. Powers, D. Brown, and M. L. Brady. The $60^\circ$ grid: Routing channels in width $d/\sqrt{3}$. In *Proceedings of the 1st Great Lakes Symposium on VLSI*, pages 214–219, Kalamazoo, Michigan, 1991.

[316] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, New York, 2nd edition, 1988.

[317] R. C. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36(6):1389–1401, 1957.

[318] H. J. Prömel and A. Steger. RNC-approximation algorithms for the Steiner problem. In *STACS'97 Proceedings*, volume 1200 of *Lecture Notes in Computer Science*, pages 559–570, 1997.

[319] H. J. Prömel and A. Steger. *The Steiner Tree Problem: A Tour Through Graphs, Algorithms, and Complexity*. Advanced Lectures in Mathematics. Friedrick Vieweg and Son, 2002.

[320] J. S. Provan. An approximation scheme for finding Steiner trees with obstacles. *SIAM Journal on Computing*, 17(5):920–934, 1988.

[321] J. S. Provan. Convexity and the Steiner tree problem. *Networks*, 18:55–72, 1988.

[322] B. Ramamurthy, J. Iness, and B. Mukherjee. Minimizing the number of optical amplifiers needed to support a multi-wavelength optical LAN/MAN. In *Proceedings of INFOCOM'97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution*, volume 1, pages 261–268, Kobe, Japan, 1997.

[323] S. K. Rao, P. Sadayappan, F. K. Hwang, and P. W. Shor. The rectilinear Steiner arborescence problem. *Algorithmica*, 7:277–288, 1992.

[324] D. Rautenbach. Rectilinear spanning trees versus bounding boxes. *The Electronic Journal of Combinatorics*, 11:N12, 2004.

[325] D. S. Richards and J. S. Salowe. A simple proof of Hwang's theorem for rectilinear Steiner minimal trees. *Annals of Operations Research*, 33:549–556, 1991.

[326] D. S. Richards and J. S. Salowe. A linear-time algorithm to construct a rectilinear Steiner minimal tree for $k$-extremal point sets. *Algorithmica*, 7(2/3):247–276, 1992.

[327] J. Riordan. The enumeration of labeled trees by degrees. *Bulletin of the American Mathematical Society*, 72(1):110–112, 1966.

[328] G. Robins and A. Zelikovsky. Minimum Steiner tree construction. In C. J. Alpert, D. P. Mehta, and S. S. Sapatnekar, editors, *Handbook of Algorithms for Physical Design Automation*, chapter 24, pages 487–508. CRC Press, Boca, Raton, 2009.

[329] J. H. Rubinstein and D. A. Thomas. The Steiner ratio conjecture for six points. *Journal of Combinatorial Theory, Series A*, 58(1):54–77, 1991.

[330] J. H. Rubinstein and D. A. Thomas. Graham's problem on shortest networks for points on a circle. *Algorithmica*, 7(1-6):193–218, 1992.

[331] J. H. Rubinstein, D. A. Thomas, and J. F. Weng. Degree-five Steiner points cannot reduce network costs for planar sets. *Networks*, 22(6):531–537, 1992.

[332] J. H. Rubinstein, D. A. Thomas, and J. F. Weng. Minimum networks for four points in space. *Geometriae Dedicata*, 93:57–70, 2002.

[333] J. H. Rubinstein, D. A. Thomas, and N. C. Wormald. Steiner trees for terminals constrained to curves. *SIAM Journal on Discrete Mathematics*, 10(1):1–17, 1997.

[334] S. M. Sait and H. Youssef. *VLSI Physical Design Automation: Theory and Practice*. World Scientific Publishing, Singapore, 1999.

[335] J. S. Salowe and D. M. Warme. Thirty-five-point rectilinear Steiner minimal trees in a day. *Networks*, 25(2):69–87, 1995.

[336] R. Samanta, A. Erzin, and S. Raha. Timing-driven Steiner tree construction on uniform $\lambda$-geometry. In *18th International Symposium on VLSI Design and Test*, pages 1–4. IEEE, 2014.

[337] D. Sankoff and P. Rousseau. Locating the vertices of a Steiner tree in an arbitrary metric space. *Mathematical Programming*, 9:240–246, 1975.

[338] M. Sarrafzadeh. *Hierarchical Approaches to VLSI Circuit Layout*. PhD thesis, University of Illinois at Urbana-Champaign, 1986.

[339] M. Sarrafzadeh, W.-L. Lin, and C. K. Wong. Floating Steiner trees. *IEEE Transactions on Computers*, 47(2):197–211, 1998.

[340] M. Sarrafzadeh and C. K. Wong. Bottleneck Steiner trees in the plane. *IEEE Transactions on Computers*, 41:370–374, 1992.

[341] M. Sarrafzadeh and C. K. Wong. Hierarchical Steiner tree construction in uniform orientations. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11:1095–1103, 1992.

[342] M. Sarrafzadeh and C. K. Wong. *An Introduction to VLSI Physical Design*. McGraw-Hill, New York, 1996.

[343] P. Saxena, R. S. Shelar, and S. Sapatnekar. *Routing Congestion in VLSI Circuits: Estimation and Optimization*. Springer, New York, 2007.

[344] R. Scheifele. Steiner trees with bounded RC-delay. In *12th Workshop on Approximation and Online Algorithms*, Wroclaw, Poland, 2014.

[345] M. Ó. Searcóid. *Metric Spaces*. Springer, New York, 2006.

[346] A. Segev. The node-weighted Steiner tree problem. *Networks*, 17(1):1–17, 1987.

[347] S. Shang, X. Hu, and T. Jing. Rotational Steiner ratio problem under uniform orientation metrics. In *Discrete Geometry, Combinatorics and Graph Theory*, volume 4381 of *Lecture Notes in Computer Science*, pages 166–176. 2007.

[348] M. Sharir and A. Schorr. On shortest paths in polyhedral spaces. *SIAM Journal on Computing*, 15(1):193–215, 1986.

[349] C. F. Shen. *The $\lambda$-geometry Steiner Minimal Tree Problem and Visualization*. PhD thesis, Northwestern University, Evanston, IL, USA, 1997.

[350] N. A. Sherwani. *Algorithms for VLSI Physical Design Automation*. Kluwer Academic Publishers, Boston, 3rd edition, 1999.

[351] A. F. Sirodenko. Minimal rectilinear Steiner trees. *Diskretnaya Matematika*, 1:28–37, 1989.

[352] J. M. Smith, Y. Jang, and M. K. Kim. Steiner minimal trees, twist angles, and the protein folding problem. *PROTEINS: Structure, Function, and Bioinformatics*, 66(4):889–902, 2007.

[353] J. M. Smith and J. S. Liebman. Steiner trees, Steiner circuits and the interference problem in building design. *Engineering Optimization*, 4:15–36, 1979.

[354] W. D. Smith. How to find Steiner minimal trees in Euclidean $d$-space. *Algorithmica*, 7(2/3):137–177, 1992.

[355] W. D. Smith and J. M. Smith. On the Steiner ratio in Euclidean 3-space. *Journal of Combinatorial Theory, Series A*, 69:301–332, 1995.

[356] T. L. Snyder. On the exact location of Steiner points in general dimension. *SIAM Journal on Computing*, 21(1):163–180, 1992.

[357] X. Song and X. Tan. An optimal channel-routing algorithm in the Times Square model. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13:891–998, 1994.

[358] J. Soukup. On minimum cost networks with nonlinear costs. *SIAM Journal on Applied Mathematics*, 29(4):571–581, 1975.

[359] J. Soukup. Circuit layout. *Proceedings of the IEEE*, 69(10):1281–1304, 1981.

[360] J. A. Stückelberger, H. R. Heinimann, W. Chung, and M. Ulber. Automatic road-network planning for multiple objectives. In *Proceedings of the 29th Council on Forest Engineering Conference*, volume 30, pages 233–248, Coeur dAlene, Idaho, 2006.

[361] K. J. Swanepoel. The local Steiner problem in normed planes. *Networks*, 36:104–113, 2000.

[362] R. Tamassia and I. G. Tollis. Planar grid embedding in linear time. *IEEE Transactions on Circuits and Systems*, 36(9):1230–1234, 1989.

[363] X. Tan and X. Song. Hexagonal 3-layer channel routing. *Information Processing Letters*, 55:223–228, 1995.

[364] X. Tan and X. Song. Routing multiterminal nets on an hexagonal grid. *Discrete Applied Mathematics*, 90:245–255, 1999.

[365] S. Teig. The X Architecture: Not your father's diagonal wiring. In *International Workshop on System-Level Interconnect Prediction (SLIP)*, pages 33–37. ACM, 2002.

[366] D. A. Thomas, M. Brazil, D. H. Lee, and N. C. Wormald. Network modelling of underground mine layout: Two case studies. *International Transactions in Operational Research*, 14(2):143–158, 2007.

[367] D. A. Thomas and J. F. Weng. Polynomial time algorithms for the rectilinear Steiner tree problem. In X. Cheng and D. Z. Du, editors, *Steiner Trees in Industry*, pages 405–426. Springer, New York, 2001.

[368] D. A. Thomas and J. F. Weng. Minimum cost flow-dependent communication networks. *Networks*, 48:39–46, 2006.

[369] C. D. Thomborson, B. Alpern, and L. Carter. Rectilinear Steiner tree minimization on a workstation. In N. Dean and G. E. Shannon, editors, *Discrete Mathematics and Theoretical Computer Science*, volume 15 of *Computational Support for Discrete Mathematics, DIMACS Series*. American Mathematical Society, Providence, Rhode Island, 1994.

[370] A. C. Thompson. *Minkowski Geometry*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, Cambridge, 1996.

[371] A. P. Thurber and G. Xue. Computing hexagonal Steiner trees using PCX. In *Proceedings of the 6th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pages 381–384, Pafos, Cyprus, 1999.

[372] I. Tomescu and M. Zimand. Minimum spanning hypertrees. *Discrete Applied Mathematics*, 54:67–76, 1994.

[373] B. Toppur and J. M. Smith. A sausage heuristic for Steiner minimal trees in three-dimensional Euclidean space. *Journal of Mathematical Modelling and Algorithms*, 4(2):199–217, 2005.

[374] E. Torricelli. De maximis et minimis. In G. Loria and G. Vassura, editors, *Opere di Evangelista Torricelli*. Faenza, Italy, 1919.

[375] E. Uchoa, M. Poggi de Aragão, and C. Ribeiro. Preprocessing Steiner problems from VLSI layout. Technical Report MCC. 32/99, PUC-Rio, Brazil, 1999.

[376] E. Uchoa, M. Poggi de Aragão, and C. Ribeiro. Preprocessing Steiner problems from VLSI layout. *Networks*, 40:38–50, 2002.

[377] A. Underwood. A modified Melzak procedure for computing node-weighted Steiner trees. *Networks*, 27:73–79, 1996.

[378] M. G. Volz. *Gradient-Constrained Flow-Dependent Networks for Underground Mine Design*. PhD thesis, The University of Melbourne, 2008.

[379] M. G. Volz, M. Brazil, C. J. Ras, K. J. Swanepoel, and D. A. Thomas. The Gilbert arborescence problem. *Networks*, 61(3):238–247, 2013.

[380] J. Vygen. *Theory of VLSI Layout*. Habilitation, University of Bonn, 2001.

[381] J. Vygen. Faster algorithm for optimum Steiner trees. *Information Processing Letters*, 111:1075–1079, 2011.

[382] W. H. Wagner. Problems in the classification of ferns. In *Recent Advances in Botany*, pages 841–844. University of Toronto Press, Toronto, 1961.

[383] J. A. Wald and C. J. Colbourn. Steiner trees, partial 2-trees, and minimum IFI networks. *Networks*, 13(2):159–167, 1983.

[384] D. C. Wang. Novel schemes for IC layout, Part I: Two-layer channel routing. In *Proceedings of the 28th ACM/IEEE Design Automation Conference*, pages 49–53, San Francisco, California, 1991.

[385] F. Wang, D. Wang, and J. Liu. Traffic-aware relay node deployment for data collection in wireless sensor networks. In *6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, 2009. (SECON '09)*, pages 1–9, Rome, Italy, 2009.

[386] L. Wang and D.-Z. Du. Approximations for a bottleneck Steiner tree problem. *Algorithmica*, 32(4):554–561, 2002.

[387] J. E. Ward and R. E. Wendell. Using block norms for location modeling. *Operations Research*, 33(5):1074–1090, 1985.

[388] D. M. Warme. *Spanning Trees in Hypergraphs with Applications to Steiner Trees*. PhD thesis, University of Virginia, 1998.

[389] D. M. Warme, P. Winter, and M. Zachariasen. Exact solutions to large-scale plane Steiner tree problems. In *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 979–980, Baltimore, Maryland, 1999.

[390] D. M. Warme, P. Winter, and M. Zachariasen. Exact algorithms for plane Steiner tree problems: A computational study. In D.-Z. Du, J. M. Smith, and J. H. Rubinstein, editors, *Advances in Steiner Trees*, pages 81–116. Kluwer Academic Publishers, Boston, 2000.

[391] D. M. Warme, P. Winter, and M. Zachariasen. GeoSteiner 3.1. Department of Computer Science, University of Copenhagen (DIKU), http://www.diku.dk/hjemmesider/ansatte/martinz/geosteiner/, 2001.

[392] A. Weber. *Über den Standort der Industrien. Teil I: Reine Theorie des Standorts*. Verlag J. C. B. Mohr, 1909. Translated by C. J. Friedrich as *Alfred Weber's Theory of the Location of Industries*. Chicago University Press, Chicago, 1929.

[393] E. Welzl. Constructing the visibility graph for $n$-line segments in $O(n^2)$ time. *Information Processing Letters*, 20:167–171, 1985.

[394] J. F. Weng. Shortest networks for smooth curves. *SIAM Journal on Optimization*, 7(4):1054–1068, 1997.

[395] J. F. Weng. Minimum networks for separating and surrounding objects. In X. Cheng and D.-Z. Du, editors, *Steiner Trees in Industry*, volume 11 of *Combinatorial Optimization*, pages 427–439. Springer, New York, 2001.

[396] J. F. Weng. Generalized Melzak's construction in the Steiner tree problem. *International Journal of Computational Geometry & Applications*, 12(06):481–488, 2002.

[397] J. F. Weng and J. M. Smith. Steiner minimal trees with one polygonal obstacle. *Algorithmica*, 29(4):638–648, 2001.

[398] G. O. Wesolowsky. The Weber problem: History and perspectives. *Location Science*, 1:5–23, 1993.

[399] D. B. West. *Introduction to Graph Theory*, volume 2. Prentice-Hall, Upper Saddle River, New Jersey, 2001.

[400] J. E. Wetzel. Converses of Napoleon's theorem. *The American Mathematical Monthly*, 99(4):339–351, 1992.

[401] T. Whitney and C. Mead. An integer based hierarchical representation for VLSI. In *Advanced Research in VLSI (Proceedings of the 4th MIT Conference)*, pages 241–257, Cambridge, Massachusetts, 1986.

[402] P. Widmayer, Y. F. Wu, and C. K. Wong. Distance problems in computational geometry with fixed orientations. In *Proceedings of the Symposium on Computational Geometry, Baltimore, MD,* pages 186–195, 1985.

[403] P. Widmayer, Y. F. Wu, and C. K. Wong. On some distance problems in fixed orientations. *SIAM Journal on Computing*, 16(4):728–746, 1987.

[404] P. Winter. An algorithm for the Steiner problem in the Euclidean plane. *Networks*, 15:323–345, 1985.

[405] P. Winter. Euclidean Steiner minimal trees with obstacles and Steiner visibility graphs. *Discrete Applied Mathematics*, 47:187–206, 1993.

[406] P. Winter. Reductions for the rectilinear Steiner tree problem. *Networks*, 26:187–198, 1995.

[407] P. Winter. Optimal Steiner hull algorithm. *Computational Geometry: Theory and Applications*, 23(2):163–169, 2002.

[408] P. Winter and J. M. Smith. Steiner minimal trees for three points with one convex polygonal obstacle. *Annals of Operations Research*, 33:577–599, 1991.

[409] P. Winter and M. Zachariasen. Euclidean Steiner minimum trees: An improved exact algorithm. *Networks*, 30:149–166, 1997.

[410] C. Witzgall. Optimal location of a central facility: Mathematical models and concept. Technical report, National Bureau of Standards, Washington, D.C., 1965.

[411] R. Wong. A dual ascent approach for Steiner tree problems on a directed graph. *Mathematical Programming*, 28:271–287, 1984.

[412] C. Wulff-Nilsen. Higher dimensional rectilinear Steiner minimal trees. Master's thesis, Department of Computer Science, University of Copenhagen, 2006.

[413] C. Wulff-Nilsen. Bounding the expected number of rectilinear full Steiner trees. *Networks*, 56(1):1–10, 2010.

[414] Y. Xin, T. Guven, and M. Shayman. Relay deployment and power control for lifetime elongation in sensor networks. In *IEEE International Conference on Communications*, volume 8, pages 3461–3466, Istanbul, Turkey, 2006.

[415] K. Xu, H. Hassanein, G. Takahara, and Q. Wang. Relay node deployment strategies in heterogeneous wireless sensor networks. *IEEE Transactions on Mobile Computing*, 9(2):145–159, 2010.

[416] G. Xue. A branch-and-bound algorithm for computing node weighted Steiner minimum trees. In T. Jiang and D. T. Lee, editors, *Computing and Combinatorics*, volume 1276 of *Lecture Notes in Computer Science*, pages 383–392. Springer, Berlin-Heidelberg, 1997.

[417] G. Xue and D.-Z. Du. An $O(n \log n)$ average time algorithm for computing the shortest network under a given topology. *Algorithmica*, 23:354–362, 1999.

[418] G. Xue, G.-H. Lin, and D.-Z. Du. Grade of service Steiner minimum trees in the Euclidean plane. *Algorithmica*, 31:479–500, 2004.

[419] G. Xue and K. Thulasiraman. Computing the shortest network under a fixed topology. *IEEE Transactions on Computers*, 51:1117–1120, 2002.

[420] G. Y. Yan, A. Albrecht, G. H. Yound, and C. K. Wong. The Steiner tree problem in orientation metrics. *Journal of Computer and System Sciences*, 55:529–546, 1997.

[421] J.-T. Yan. An improved optimal algorithm for bubble-sorting-based non-Manhattan channel routing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 18:163–171, 1999.

[422] J.-T. Yan. Timing-driven octilinear Steiner tree construction based on Steiner-point reassignment and path reconstruction. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 13(2):26, 2008.

[423] Y. Y. Yang and O. Wing. An algorithm for the wiring problem. In *Digest of the IEEE International Symposium on Electrical Networks*, pages 14–15, 1971.

[424] Y. Y. Yang and O. Wing. Suboptimal algorithm for a wire routing problem. *IEEE Transactions on Circuit Theory*, 19(5):508–510, 1972.

[425] Y. Y. Yang and O. Wing. On a multinet wiring problem. *IEEE Transactions on Circuit Theory*, 20(3):250–252, 1973.

[426] M. C. Yildiz and P. H. Madden. Preferred direction Steiner trees. In *Proceedings of the 11th ACM Great Lakes Symposium on VLSI (GLSVLSI)*, pages 56–61, West Lafayette, Indiana, 2001.

[427] M. C. Yildiz and P. H. Madden. Preferred direction Steiner trees. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 21:1368–1372, 2002.

[428] M. Yue. A report on the Steiner ratio conjecture. *OR Translations*, 4:1–21, 2000.

[429] M. Zachariasen. Rectilinear full Steiner tree generation. *Networks*, 33:125–143, 1999.

[430] M. Zachariasen. A catalog of Hanan grid problems. *Networks*, 38:76–83, 2001.

[431] M. Zachariasen. The rectilinear Steiner tree problem: A tutorial. In D.-Z. Du and X. Cheng, editors, *Steiner Trees in Industries*, pages 467–507. Kluwer Academic Publishers, Boston, 2001.

[432] M. Zachariasen. Comment on "Computing the shortest network under a fixed topology". *IEEE Transactions on Computers*, 55:783–784, 2006.

[433] M. Zachariasen. Fixed orientation interconnection problems: Theory, algorithms and applications. Technical report, University of Copenhagen (Doctor of Science Dissertation), 2009.

[434] M. Zachariasen and A. Rohe. Rectilinear group Steiner trees and applications in VLSI design. *Mathematical Programming*, 94:407–433, 2003.

[435] M. Zachariasen and P. Winter. Obstacle-avoiding Euclidean Steiner trees in the plane: An exact algorithm. In *Workshop on Algorithm Engineering and Experimentation (ALENEX), Lecture Notes in Computer Science 1619*, pages 282–295, Baltimore, Maryland, 1999. Springer.

# Index