Summer 2010

Exact and heuristic algorithms for the Euclidean Steiner tree problem

Jon William Van Laarhoven
University of Iowa

EXACT AND HEURISTIC ALGORITHMS FOR THE EUCLIDEAN STEINER TREE PROBLEM

by

Jon William Van Laarhoven

An Abstract

Of a thesis submitted in partial fulfillment of the
requirements for the Doctor of Philosophy degree in
Applied Mathematical and Computational Sciences in
the Graduate College of
The University of Iowa

July 2010

Thesis Supervisors: Professor Kurt Anstreicher
Associate Professor Jeffrey Ohlmann

ABSTRACT

In this thesis, we study the Euclidean Steiner tree problem (ESTP) which arises in the field of combinatorial optimization. The ESTP asks for a network of minimal total edge length spanning a set of given terminal points in $\Re^d$ with the ability to add auxiliary connecting points (*Steiner points*) to decrease the overall length of the network. The graph theory literature contains extensive studies of exact, approximation, and heuristic algorithms for ESTP in the plane, but less is known in higher dimensions. The contributions of this thesis include a heuristic algorithm and enhancements to an exact algorithm for solving the ESTP.

We present a local search heuristic for the ESTP in $\Re^d$ for $d \geq 2$. The algorithm utilizes the Delaunay triangulation to generate candidate Steiner points for insertion, the minimum spanning tree to create a network on the inserted points, and second order cone programming to optimize the locations of Steiner points. Unlike other ESTP heuristics relying on the Delaunay triangulation, the algorithm inserts Steiner points probabilistically into Delaunay triangles to achieve different subtrees on subsets of terminal points. The algorithm extends effectively into higher dimensions, and we present computational results on benchmark test problems in $\Re^d$ for $2 \leq d \leq 5$.

We develop new geometric conditions derived from properties of Steiner trees which bound below the number of Steiner points on paths between terminals in the Steiner minimal tree. We describe conditions for trees with a Steiner topology and show how these conditions relate to the Voronoi diagram. We describe more restrictive conditions for trees with a full Steiner topology and their implementation into the algorithm of Smith (1992). We present computational results on benchmark test problems in $\Re^d$ for $2 \leq d \leq 5$.

Abstract Approved: _____
Thesis Supervisor

_____
Title and Department

_____
Date

_____
Thesis Supervisor

_____
Title and Department

_____
Date

EXACT AND HEURISTIC ALGORITHMS FOR THE EUCLIDEAN STEINER TREE PROBLEM

by

Jon William Van Laarhoven

A thesis submitted in partial fulfillment of the
requirements for the Doctor of Philosophy degree in
Applied Mathematical and Computational Sciences in
the Graduate College of
The University of Iowa

July 2010

Thesis Supervisors: Professor Kurt Anstreicher
Associate Professor Jeffrey Ohlmann

Graduate College
The University of Iowa
Iowa City, Iowa

CERTIFICATE OF APPROVAL

PH.D. THESIS

This is to certify that the Ph.D. thesis of

Jon William Van Laarhoven

has been approved by the Examining Committee
for the thesis requirement for the Doctor of
Philosophy degree in Applied Mathematical and
Computational Sciences at the July 2010 graduation.

Thesis Committee: _____
Kurt Anstreicher, Thesis Supervisor

_____
Jeffrey Ohlmann, Thesis Supervisor

_____
Samuel Burer

_____
Yi Li

_____
Kasturi Varadarajan

ABSTRACT

In this thesis, we study the Euclidean Steiner tree problem (ESTP) which arises in the field of combinatorial optimization. The ESTP asks for a network of minimal total edge length spanning a set of given terminal points in $\Re^d$ with the ability to add auxiliary connecting points (*Steiner points*) to decrease the overall length of the network. The graph theory literature contains extensive studies of exact, approximation, and heuristic algorithms for ESTP in the plane, but less is known in higher dimensions. The contributions of this thesis include a heuristic algorithm and enhancements to an exact algorithm for solving the ESTP.

We present a local search heuristic for the ESTP in $\Re^d$ for $d \geq 2$. The algorithm utilizes the Delaunay triangulation to generate candidate Steiner points for insertion, the minimum spanning tree to create a network on the inserted points, and second order cone programming to optimize the locations of Steiner points. Unlike other ESTP heuristics relying on the Delaunay triangulation, the algorithm inserts Steiner points probabilistically into Delaunay triangles to achieve different subtrees on subsets of terminal points. The algorithm extends effectively into higher dimensions, and we present computational results on benchmark test problems in $\Re^d$ for $2 \leq d \leq 5$.

We develop new geometric conditions derived from properties of Steiner trees which bound below the number of Steiner points on paths between terminals in the Steiner minimal tree. We describe conditions for trees with a Steiner topology and show how these conditions relate to the Voronoi diagram. We describe more restrictive conditions for trees with a full Steiner topology and their implementation into the algorithm of Smith (1992). We present computational results on benchmark test problems in $\Re^d$ for $2 \leq d \leq 5$.

TABLE OF CONTENTS

LIST OF TABLES

Table

LIST OF FIGURES

Figure

CHAPTER 1
INTRODUCTION

The Euclidean Steiner tree problem (ESTP) arises in the field of combinatorial optimization; the problem and its variations find vast applications in the study of networks (Du and Hu, 2008). The ESTP is simple to state, difficult to solve, and unlike many other combinatorial optimization problems, has an additional geometric component. This chapter provides historical context, definitions, and a discussion of the computational intractability of the ESTP and a few variant problems.

1.1 Historical background

The root of the Euclidean Steiner tree problem (ESTP) dates back to Fermat (1601-1665) who proposed the following problem in a letter: "Given three points in the plane, find a fourth point such that the sum of its distances to the three points is a minimum." This problem is often called the *Fermat problem*, and it captured the interest of prominent mathematicians including Torricelli, Cavalieri, and Simpson. Torricelli proposed a geometric solution to this problem as follows: place equilateral triangles on the sides of and outside of $\triangle pqr$, and construct circles circumscribing each of the three equilateral triangles. Torricelli claimed that the three circles intersect at the point s which minimizes the sum of distances. This point is often called the *Torricelli point*, see Figure 1.1. However, in the case that $\triangle pqr$ contains an angle greater than $120\,^{\circ}$, the Torricelli point lies outside the triangle and is no longer the minimizing point. In this case, the minimizing point is the vertex at the obtuse angle (Kuhn, 1974).

In his *Exercitations Geometricae*, Cavalieri showed that the line segments from the three points to the Torricelli point create three equal angles of $120\,^{\circ}$. Simpson in his *Application of Fluxions* showed that the three line segments created by connecting the vertices of the equilateral triangles previously described to the

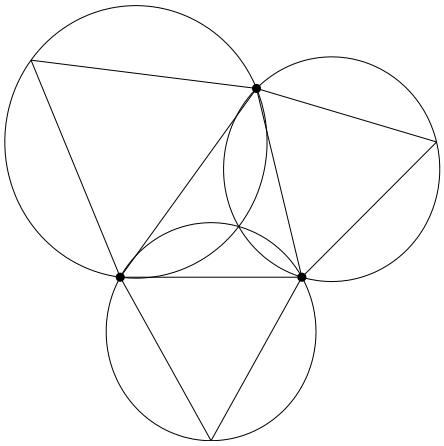


Figure 1.1: The three circumcircles of the constructed equilateral triangles intersect at the Torricelli point.

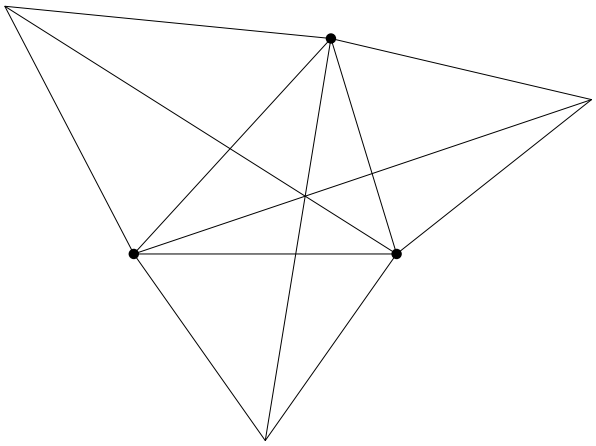


Figure 1.2: The Simpson lines intersect at the Torricelli point.

opposite vertex of $\triangle pqr$ intersected in the Torricelli point, see Figure 1.2. Courant and Robbins (1941) popularized the generalized version of the problem ($n > 3$) in their book "What is Mathematics." They dubbed the problem, the "the Steiner problem" after the mathematician Jacob Steiner (1796-1863) who considered the generalized version of the Fermat problem.

1.2 Definitions and properties

The objective of the Euclidean Steiner tree problem (ESTP) is to determine the minimal length tree (with respect to the Euclidean metric) spanning a set of *terminal points*, X, while permitting the introduction of extra points (composing a set S of *Steiner points*) into the network to reduce its overall length.

A *topology* is a configuration of terminal points and Steiner points where the connections are specified, but the locations of the Steiner points are not. A topology is said to be a *Steiner topology* if every Steiner point has degree three and every terminal point has degree at most three. A Steiner topology where all terminal points are degree one is a *full Steiner topology* (FST). A full Steiner topology on n terminal points has $n - 2$ Steiner points (Gilbert and Pollak, 1968). Any non-full Steiner topology can be identified with a FST where some edges have zero length (Gilbert and Pollak, 1968); such a FST is called *degenerate*. A solution to the ESTP is called a *Steiner minimal tree* (SMT). Gilbert and Pollak (1968) show that the SMT has a Steiner topology. In the search for minimal trees, we restrict our search to trees with a Steiner topology.

The algorithms we describe rely on properties of SMTs. Proofs of these and other properties can be found in Gilbert and Pollak (1968). The *angle condition* states that each pair of edges in a SMT meets at an angle of no less than 120°. The *degree condition* states that the degree of each terminal point is at most three and the degree of each Steiner point is exactly three. Together, these conditions imply that the three edges incident to a Steiner point meet at exactly 120°. Additional properties of Steiner trees are covered in Chapter 5.

1.3 The Steiner ratio

The minimum spanning tree (MST) is the shortest network spanning a set of points without the use of additional points (i.e., Steiner points). The MST serves as an efficient approximation algorithm for the SMT, i.e., the length of the MST

is within a known bound of the length of the SMT. Let X be a set of points in a metric space, and let $MST(X)$ and $SMT(X)$ denote the minimum spanning tree and Steiner minimal tree of the set X, respectively. Let $\ell(T)$ be the length of the network T. The *Steiner* ratio, ρ, is the minimum value of the ratio

$$\frac{\ell(SMT(X))}{\ell(MST(X))}$$

taken over all sets X. Clearly $\rho \leq 1$ since the MST can be regarded as a fully-degenerate SMT ($S = \emptyset$). Cieslik (2001), in his treatment of the Steiner ratio, proves that for any metric space $\rho \geq 1/2$. Beardwood et al. (1959) provided the first lower bound on the Steiner ratio for the planar ESTP, $\rho > 1/2$. Du and Hwang (1992) prove that for the planar ESTP, $\rho = \sqrt{3}/2 \approx 0.866$, and equality is obtained for a network of terminal points located at the vertices of an equilateral triangle. In higher dimensions, the the Steiner ratio less than $\sqrt{3}/2$ but is not known. Du and Smith (1994) disprove the conjecture of Gilbert and Pollak (1968) that the Steiner ratio is obtained for terminals at the vertices of a d-dimensional regular simplex, for $d \geq 3$. Toppur and Smith (2005) conjecture the optimal Steiner ratio in $\Re^3$ is not achieved by a single simplex, but rather by placing terminal points at the vertices of a $\Re$-sausage, an object created by linking an infinite number of regular tetrahedral simplices. Toppur and Smith (2005) empirically calculate the Steiner ratio in $\Re^3$ to be approximately 0.784190.

1.4 Computational intractability

In order to discuss the difficulty of the ESTP, we first quantify what it means for an optimization problem to be "difficult." Computational complexity theory classifies problems based on measurements of difficulty in computing solutions. A problem can be *solved in polynomial time* if, given an input of size s, there is a polynomial function, p, such that the number of operations required to solve the problem is on the order of $p(s)$. The class of all such problems is denoted by P. If

we are able to verify a proposed solution to a problem in polynomial time we say the problem is in NP, short for *non-deterministic polynomial time*. A problem which is solvable in polynomial time allows verification of solutions in polynomial time, so $P \subset NP$. A famous and long-standing problem is to determine whether $P = NP$. A (yes/no) decision problem X is *NP-complete* if i) $X \in NP$ and ii) all problems $Y \in NP$ polynomially reduce to X (roughly speaking, X is at least as difficult as the problems in NP). A problem is said to be *NP-hard* if condition ii) holds.

The ESTP is a difficult combinatorial optimization problem; Garey et al. (1977) show that decision version of the ESTP is NP-hard, but the ESTP itself is not known to be in NP. Garey et al. (1977) construct a polynomial transformation T which converts an instance of "exact cover by 3-sets" (X3C) into a planar instance of the ESTP. The transformation converts (yes/no) instances of X3C to (yes/no) instances of a Steiner tree decision problem, respectively. By exploiting geometric properties of Steiner trees and use of a tool called a "probe" (based on the angle property), it is shown this constructed Steiner instance has a *very* constrained structure, and accordingly, its length can be bounded depending on the satisfiability of the X3C instance. Since X3C is known to be NP-complete, this means the Steiner problem is at least as difficult as X3C, i.e. it is NP-hard.

The difficulty in solving the ESTP can be partially attributed to the tremendous number of FSTs that must be checked in the search for the optimal solution. The number of FSTs grows super-exponentially with respect to the the number of terminal points (Gilbert and Pollak, 1968); see Table 1.1 which illustrates how the number of FSTs grows compared against the exponential function 2^n.

1.5 Variant problems

We mention several ESTP variants which can arise from application of ESTP under different "real world" constraints. Computer chip layout design may utilize the L_1 or "taxi-cab" metric in lieu of the standard L_2 metric giving the rectilinear

Table 1.1: The number of FSTs grows super-exponentially with respect to instance size.

n	2^n	$FST(n)$
1	2	0
2	4	0
3	8	1
4	16	3
5	32	15
6	64	105
7	128	945
8	256	10,395
9	512	135,135
10	1,024	2,027,025
11	2,048	34,459,425
12	4,096	654,729,075
13	8,192	13,749,310,575
14	16,384	316,234,143,225
15	32,768	7,905,853,580,625
16	65,536	213,458,046,676,875

Steiner tree problem (RSTP). Hanan (1966) was one of the first to address the RSTP. It is known that all Steiner points lie in what is called the *underlying grid* (also called the *planar grid* or *Hanan grid*) which is the set of points created by the intersections of horizontal and vertical lines through all terminal points points (Hanan, 1966). Garey and Johnson (1977) show that RSTP is NP-hard.

Considering connecting facilities located far apart on the earth gives rise to the Steiner tree problem on the sphere. The geodesic minimum spanning tree problem (GMST) and geodesic Steiner minimal tree problem (GSMT) are analogues of the MST and SMT on the sphere, respectively. The metric adopted on the sphere is typically the "great circle" distance, where a circle on the surface of the sphere is a great circle if it lies in a plane containing the center of the sphere. The distance between points x_i and x_j on the sphere is the length of the arc of the great circle containing them. Cockayne (1972) provides analogues (as in Section 1.1) for locating a single Steiner point connected to three terminals on the unit sphere. Zhang (2003) considers the problem of n terminals and one Steiner point on the sphere. Dolan et al. (1991) provide algorithms for computing the Delaunay triangulation, Voronoi diagram, MST, and sub-optimal SMTs on the sphere.

A constrained engineering problem may require a pre-specified set of potential Steiner points and connections; this is the Steiner tree problem in networks (NSTP) also called the Steiner tree problem in graphs (Koch and Martin, 1998). The NSTP consists of an edge-weighted graph $G = (V, E, w)$ where a list of vertices V and a subset of required vertices (terminals) $R \subset V$ are specified; edge weights may be negative. The goal is to find a subtree of V which spans all points in R and is of minimal weight. On the one hand, the NSTP appears more tractable than the ESTP as the locations of all potential Steiner points are pre-specified. On the other hand, edge weights in the NSTP may be negative and there is no underlying metric space. The RSTP can be considered a special case of NSTP, as all potential Steiner

points lie in the Hanan grid. The NSTP is shown to be NP-hard (Karp, 1972).

The prize collecting Steiner tree problem (PCSTP) (a variation of the NSTP) occurs in fiber-optic network design where including customers into the network generates a reward (Ljubić et al., 2005). In the PCSTP, a set of potential vertices V is specified ($R = \emptyset$), each with an associated profit. As in the NSTP, edge weights can be negative. The objective is to minimize the sum of the total cost of all edges in the subtree plus the total profit of all vertices not contained in the subtree. The NSTP is a particular case of PCSTP, where terminals have infinite profit and optional points have no associated profit. As a result, the PCSTP is also NP-hard. Other variants of the NSTP include the node-weighted Steiner tree problem and prize collecting node-weighted Steiner tree problem (Du and Hu, 2008).

CHAPTER 2
MATHEMATICAL PRELIMINARIES

This chapter presents concepts from computational geometry, conic optimization, and implicit enumeration which we utilize in the thesis. The algorithms in Chapters 4 and 5 utilize the minimum spanning tree, Delaunay triangulation, and Voronoi diagram in their search for Steiner trees. Algorithms for solving the ESTP rely on a method of optimizing the location of Steiner points under a specified topology, we describe these methods. We discuss a general branch-and-bound framework which governs exact algorithms for solving the ESTP.

2.1 The minimum spanning tree

The Steiner minimal tree is closely related to the minimum spanning tree (MST): the shortest network connecting a set of points without the addition of Steiner points. Many heuristic and exact algorithms, including the algorithms we develop in Chapters 4 and 5, utilize the MST in the search for the SMT. For a set of points, X, with the set of Steiner points, S, inserted at the proper location, the SMT is found by forming the MST on the set $(X \cup S)$.

Prim's algorithm, Kruskal's algorithm, and the reverse-edge delete algorithm (essentially Kruskal's in reverse) are all polynomial time algorithms for computing the MST (Kleinberg and Tardos, 2005). The algorithms in Chapters 4 and 5 utilize Prim's algorithm for computing the MST, see pseudo-code in Algorithm 2.1. Prim's algorithm on n points and m edges runs in $\mathcal{O}(n^2)$ time when searching an adjacency matrix. The run-time can be improved to $\mathcal{O}(m \log n)$ time by using a heap-based priority queue, and even faster MST implementations exist (Kleinberg and Tardos, 2005).

The MST provides upper bounds on lengths of edges present in the SMT. The *bottleneck distance* denoted by b, is the length of the longest edge in the MST. Given

Algorithm 2.1 Prim's algorithm for computing the MST

1: **Input:** A set of points, X, and edge weights, A.
2: **Output:** A tree, T, on X.
3: **Initialization:** $V_{new} = x_i$ where $x_i \in X$, $E_{new} = \{\}$.
4: **while** $V_{new} \neq X$ **do**
5: Choose edge (x_i, x_j) with minimal weight s.t. $x_i \in V_{new}$ and $x_j \in X - V_{new}$
6: Set $V_{new} = V_{new} \cup x_j$ and $E_{new} = E_{new} \cup (x_i, x_j)$.
7: **end while**
8: Output E_{new}, a minimum spanning tree.

two points x_i and x_j connected in the MST, the *bottleneck distance between* x_i *and* x_j, denoted b_{ij}, is the length of the longest edge on the (unique) path between x_i and x_j in the MST.

Lemma 2.1. *A SMT contains no edge of length greater than* b.

Proof. Assume to the contrary that some edge, e, in a SMT, T, is longer than b. Remove edge e separating T into two connected components, each containing at least one terminal. These two components may reconnected with a terminal-to-terminal edge from a MST with length at most b, which shortens T and contradicts the assumption that T is a SMT. □

We sharpen the result by considering bottleneck distances between pairs of points.

Lemma 2.2. *A SMT contains no edge of length greater than* b_{ij} *on the unique path between* x_i *and* x_j.

Proof. Assume to the contrary that some edge, e, on the path between x_i and x_j in a SMT, T, is longer than b_{ij}. Remove e, separating T into two connected components one with terminal x_i (call this component C_1), one with terminal x_j (call this component C_2). Consider the path connecting terminals x_i and x_j in the MST. Beginning at terminal x_i, follow edges in the MST along the path to x_j and stop before leaving terminals in the component C_1. The next terminal-to-terminal

edge, $\tilde{e}$, may be added to T, reconnecting components C_1 and C_2. The length of $\tilde{e}$ is less than or equal to b_{ij}, which is also less than the length of e by assumption. Adding $\tilde{e}$ will only shorten the length of T, contradicting the assumption that T is a SMT. □

In order to decrease upper bounds on lengths of edges, we ask if another spanning tree can decrease the bottleneck distance between pairs of points. In fact, a MST minimizes all pairwise bottleneck distances.

Lemma 2.3. *A MST minimizes the bottleneck distance between all pairs of points x_i and x_j.*

Proof. By way of contradiction, assume there is a tree T^* so that the bottleneck distance between points x_i and x_j in T^* (call this value b^*_{ij}) is less than b_{ij}. Find the edge in a MST, T, between x_i and x_j which is longer than b^*_{ij} and remove it, separating the tree into two connected components. These two components may be re-connected with an edge on the unique path between x_i and x_j in T^*, thereby shortening T. This contradicts the assumption that T is a MST. □

This proof is illustrated in the construction of a MST in Algorithm 2.1, and in particular on line 5. The algorithm always adds the shortest edge possible, provided cycles are not created. Thus if T^* had a shorter edge, the MST algorithm would add it unless a cycle was created. If a cycle was created, there already exists a path between x_i and x_j, all of whose edges are shorter than b^*_{ij}.

2.2 The Delaunay triangulation and Voronoi diagram

Both the heuristic approach and exact algorithm we discuss in Chapters 4 and 5 rely on a notion of "closeness" between terminal points. Structures from computational geometry quantify this notion.

For Z, a set of points in $\Re^d$, the *Voronoi diagram* of Z is determined by partitioning Euclidean space into convex polyhedra (called *Voronoi regions*) with

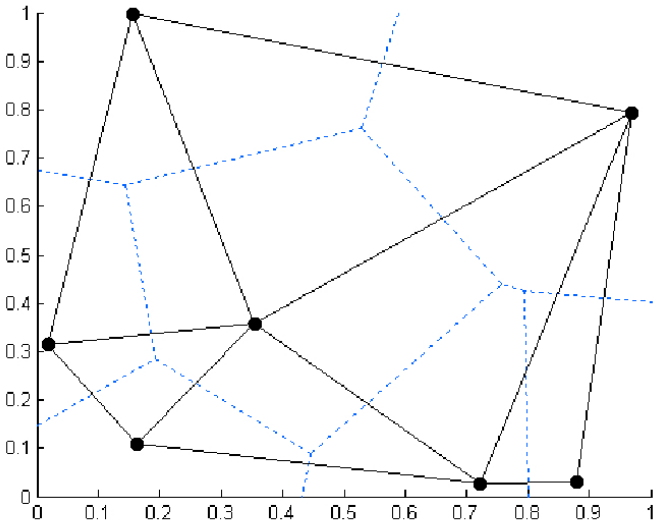


Figure 2.1: Voronoi diagram (dashed) and Delaunay triangulation (solid) in the plane.

one point in each region. For every point $z \in Z$, the Voronoi region about z denoted $vor(z)$ consists of all points closer to z than to any other point in Z. That is $vor(z) = \{u : ||u - z|| < ||u - y|| \ \forall y \in Z\}$.

The *Delaunay triangulation* is the straight-line dual graph of the Voronoi diagram, constructed by connecting two points with an edge if and only if they share a face in the Voronoi diagram. For Z, a set of points in $\Re^d$, the Delaunay triangulation, $DT(Z)$, is a subdivision of the convex hull of Z into d-dimensional simplices such that: (i) any two simplices in $DT(Z)$ intersect in a common face or not at all, (ii) Z corresponds to the set of points that are vertices of the subdividing simplices, and (iii) no $z \in Z$ is inside the circum-hypersphere of any simplex in $DT(Z)$ (De Berg et al., 2008). Figure 2.1 illustrates a Voronoi diagram and Delaunay triangulation for a set of points in the plane.

The edges of the Delaunay triangulation for a set of vertices contain the MST; there are other triangulations (Gabriel graph, relative neighborhood graph) that also contain the MST (De Berg et al., 2008).

2.3 Conic optimization

For a given topology, the location of Steiner points must be determined in order to compute the length of the tree. The algorithm for doing so can be called a *relatively minimal tree* (RMT) algorithm. The constructions highlighted in Section 1.1 allow the optimization of one Steiner point connected to three terminals. Extending the idea to multiple Steiner points, one option for a RMT algorithm is to iteratively optimize the locations of Steiner points, treating neighbors as fixed. Smith (1992) shows that this process converges to the globally optimal locations for a given topology.

A more direct method of optimizing Steiner point locations than iterative optimization comes from the idea of optimization over cones. To explain conic optimization's relation to ESTP, we begin by stating a linear optimization problem over a naturally-arising cone: the non-negative orthant. This gives the standard representation of a linear program:

$$\begin{aligned} min \quad & c^T x \\ s.t. \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

where $x \in \Re^n$ represents a variable to be determined, $b, c \in \Re^n$ represent vectors of coefficients, and A is a matrix of coefficients. The set of $x \geq 0$ for which $Ax = b$ represents the feasible region, and is known to be a convex polytope.

If we allow a convex, non-linear objective function, we can formulate the problem optimizing the location of one Steiner point incident to three terminals as the following minimum-sum-of-norms problem (a single-facility location problem):

$$\begin{aligned} min \quad & ||s_1|| + ||s_2|| + ||s_3|| \\ s.t. \quad & s_i = x_i - y, \qquad i = 1, 2, 3 \end{aligned}$$

where y is the coordinates of the Steiner point, and x_i are the coordinates of each terminal. The variables s_i represent the edges of the tree, translated with one endpoint at the origin.

It is possible to formulate the ESTP as a minimum sum-of-norms problem (Dreyer and Overton, 1998), and solve this formulation via interior point methods (Andersen et al., 2001). Consider an instance of m Steiner points and $m+2$ terminal points in $\Re^d$. The resulting FST has $n = (2m + 1)$ edges. Let $Y_{d\times m}$ be the matrix of Steiner point coordinates. Define matrices $A_{m\times n}$ and $C_{d\times n}$ as follows: if edge i is connected to a Steiner point j and a terminal point, then column c_i contains the coordinates of the adjacent terminal point and column $a_i = e_j$, where $e_j \in \Re^m$ is the j^{th} elementary column vector. If edge i is adjacent to two Steiner points j and k with $j < k$, then $c_i = 0$ and $a_i = e_k - e_j$. We can then express the problem of locating the Steiner points for a given full Steiner topology as a minimum sum of norms problem:

$$min \quad \sum_{i=1}^{n} ||s_i|| \tag{2.1}$$

$$s.t. \quad S = C - YA \tag{2.2}$$

where $S_{d\times n}$ is a matrix whose i^{th} column, s_i, represents edge i in the tree translated to the origin.

The minimum sum-of-norms problem has a convex but non-differentiable objective function which presents computational challenges for interior point methods. Second order cone programming (SOCP), a generalization of linear programming based on the idea of optimizing a linear function over the intersection of an affine set and the product of second-order (quadratic) cones, overcomes this difficulty by embedding the non-differentiable objective function into the constraints. Formulating the minimum sum-of-norms problem as an SOCP allows the use of interior point methods (Nesterov and Nemirovsky, 1994) or even more powerful primal-dual

methods (Nesterov and Todd, 1998).

We introduce some definitions to develop the SOCP formulation. Given a real variable x and a d-dimensional vector v, we say $\begin{bmatrix} x \\ v \end{bmatrix}$ is in a $d+1$ dimensional second order (quadratic) cone if $x \geq ||v||$, denoted $\begin{bmatrix} x \\ v \end{bmatrix} \geq_Q 0$. The cones correspond to the non-negative orthant in linear programming, and so linear programming is a special case of SOCP. We re-formulate the problem of optimizing one Steiner point incident to three terminals from a minimum sum-of-norms problem into an SOCP by introducing a $d+1$ dimensional cone for each edge:

$$\begin{aligned} min \quad & \sigma_1 + \sigma_2 + \sigma_3 \\ s.t. \quad & \sigma_i \geq ||x_i - y||, \qquad i = 1, 2, 3 \end{aligned}$$

which we expand to delineate SOCP's relation to a linear program:

$$\begin{aligned} min \quad & \sigma_1 + \sigma_2 + \sigma_3 \\ s.t. \quad & s_i = x_i - y, \qquad i = 1, 2, 3 \\ & \begin{bmatrix} \sigma_i \\ s_i \end{bmatrix} \geq_Q 0, \qquad i = 1, 2, 3 \end{aligned}$$

where y is the coordinates of the Steiner point, and x_i are the coordinates of each terminal. If each x_i or y is a vector in $\Re^d$, each edge of the tree generates a $(d+1)$-dimensional cone. For each value of σ_i, the cone is defined by the set of all y for which $\sigma_i \geq ||x_i - y||$.

Lastly, we reformulate the problem of optimizing Steiner points for a given FST into an SOCP as follows, where c_i and y are defined as in the minimum sum-of-norms problem in equation 2.2:

$$\begin{aligned} min \quad & \sum_{i=1}^{n} \sigma_i \\ s.t. \quad & s_i = c_i - A_i^T y, \qquad i = 1, ..., n \\ & \begin{bmatrix} \sigma_i \\ s_i \end{bmatrix} \geq_Q 0, \qquad i = 1, ..., n \end{aligned}$$

where $A_i^T = a_i^T \otimes I_d$, I_d is a d-by-d identity matrix, and $\otimes$ denotes the Kronecker product of two matrices. For a 1-by-n matrix a_i^T and a d-by-d matrix I_d, $a_i^T \otimes I_d$

denotes the d-by-(dn) block matrix:

$$\begin{pmatrix} a_1 I_d & \cdots & a_n I_d \end{pmatrix}.$$

2.4 Branch-and-bound enumeration

Algorithms which solve the ESTP to optimality must verify that the proposed solution is truly optimal; to do so, all potential solutions must be checked. As highlighted in Table 1.1, it would require a significant amount of time to consider all the FSTs by brute-force; a more organized method of enumerating FSTs is needed. This motivates the concept of *implicit enumeration* where the entire solution space S is considered, but many elements of S are not explicitly computed. In a 0-1 integer program, for example, if it is known that the m^{th} entry of the solution vector must be 1, then the size of the search space is halved by omitting from consideration any solution with 0 in the m^{th} entry.

The exact algorithms for ESTP described in Section 3.3 operate within a branch-and-bound framework. *Branch-and-bound* (BB) is an implicit enumeration technique which excludes portions of the solution space based on values of best-found solutions. A branch-and-bound algorithm has three main components: a bounding function, a strategy for selecting the next node (element of the search space), and a branching rule. For a minimization problem, the bounding function, B, provides a lower bound for the best obtainable solution in a subspace, P. If P is a leaf of the branch-and-bound tree, then $B(P) = v(P)$ where $v(P)$ denotes the correct solution value for the problem P. The node selection strategy decides the next node to be processed. A "depth-first" selection strategy chooses nodes to be processed which are deeper in the enumeration tree. Upon reaching a leaf of the tree, the algorithm backtracks. A "breadth-first" selection strategy explores all nodes at a given level before processing any nodes at the next level. A breadth-first strategy may require too many nodes to be stored in memory, so depth-first search is often used in practice. The branching rule applies if the current node cannot

be removed from consideration. The subspace is divided into further subspaces (children) which are added to a list to be explored in later iterations. We include basic branch-and-bound pseudo code in Algorithm 2.4, and we incorporate a "lazy" node selection strategy whereby the active nodes are stored with the bound of their parent.

As an example, consider an optimization problem with n binary variables, $x_i = \{0, 1\}$, for $i = 1, ..., n$. When a node is processed and not fathomed, the typical branching rule creates two children by setting $x_j = 0$ for one child, and $x_j = 1$ for the other child (for some $1 \leq j \leq n$). The bounding function, B, often allows the remaining binary variables to be continuous on [0,1], and possibly allows additional constraints that are known to be valid for binary values of those variables. The complete enumeration tree has 2^n leaves, but hopefully far fewer problems are considered when using the branch-and-bound framework. For further reading on branch-and-bound enumeration, see Clausen (1997).

Algorithm 2.2 Lazy branch and bound for a minimization problem

1: **Initialization:** $Best = \infty, List = \{(RootNode, \infty)\}$.
2: Repeat until $List = \emptyset$:
3: Remove a node P from $List$ to be made active;
4: Can P be fathomed from parent bound? If yes, fathom P and Go to line 3;
5: Determine bound $B(P)$ for P. If $B(P) >= UB$, fathom P and Go to line 3;
6: If P is a leaf of the tree, update $UB = B(P)$; Go to line 3;
7: If P is not a leaf of the tree, create children of P, add to $List$ along with bound $B(P)$. Go to line 3.

CHAPTER 3
SOLUTION METHODS FOR THE ESTP

In solving the ESTP, a choice must be made between solution quality versus computational effort. An exact algorithm can be used to completely explore the solution space and return the provably optimal solution. An approximation algorithm may be used if a solution within a fixed percent of optimal is required. If computational effort is important, a heuristic may be used to return a (hopefully good quality) solution in a reasonable computational time. This chapter overviews these three solution methods in the context of the ESTP.

3.1 Approximation algorithms

An *approximation algorithm* produces solutions within a guaranteed percentage of the optimal solution; such an algorithm which runs in polynomial time (with respect to instance size) is called a *polynomial time approximation scheme* (PTAS). For example, a MST-producing algorithm (such as Prim's algorithm) can be viewed as a PTAS for ESTP due to the Steiner ratio. The strongest approximation algorithms produce solutions within a factor of $(1 + 1/c) * optimal$, $c > 0$ so that solutions of arbitrarily high quality may be obtained (although at the cost of increased computation time). Arora (1998) shows that many NP-hard geometric optimization problems (ESTP, rectilinear Steiner tree problem, traveling salesman problem) all have a $(1 + 1/c)$-PTAS. Using recursive partitioning, "portals" to connect created subproblems, and dynamic programming, Arora (1998) develops a $(1 + 1/c)$-PTAS that runs in $n(\log n)^{\mathcal{O}(c\sqrt{d})^{d-1}}$ time on instances with n terminal points in $\Re^d$. Rao and Smith (1998) utilize "spanners" and "banyans" in lieu of portals to develop a faster PTAS for ESTP. They describe a (1+1/c)-PTAS which runs in $2^{(cd)^{\mathcal{O}(d)}}n + (cd)^{\mathcal{O}(d)}n \log n$ time on instances with n terminal points in $\Re^d$.

3.2 Heuristic algorithms

Section 1.4 outlines the computational intractability of the ESTP: solving the problem to optimality requires significant computational effort, particularly on large instances. This motivates the use of a heuristic algorithm which finds a good solution with moderate computational effort. This section gives definitions related to local search heuristics and reviews the literature on heuristic search for the ESTP.

3.2.1 Local search

A *local search* heuristic produces an optimal or near-optimal solution by making iterative changes to an initial solution. Local search algorithms have three primary features: solution representation, solution evaluation, and neighbor solution generation. Recall that the objective of the ESTP is to find a minimal length Steiner tree spanning the set of specified terminal points. Evaluating a solution for the ESTP requires a relatively minimal tree (RMT) algorithm (which may vary across heuristics) to optimize the locations of Steiner points in the solution's pre-specified topology. The solution representation and neighbor generation mechanisms vary across heuristics, and we use these differences as a classification scheme for ESTP heuristics.

3.2.2 Planar heuristics

The literature abounds with heuristics for ESTP in the plane, which we outline based on solution representation and neighbor generation schemes. Most planar heuristics operate by either i) inserting a candidate set of Steiner points into the network and forming a Steiner topology on this augmented set of points or ii) finding good FSTs on subsets of terminal points and concatenating these partial FSTs into a Steiner tree.

3.2.2.1 Insertion heuristics

We first discuss Steiner point insertion heuristics, as the heuristic algorithm we develop in Chapter 4 can be classified as such an approach. Given the optimal number and location of all Steiner points, the SMT can be formed by constructing the MST of $X \cup S$, the union of the set of terminal points and Steiner points. This relationship between the MST and SMT motivates ESTP heuristics which generate candidate solutions by inserting a set S of Steiner points into a topology and then form the MST of $X \cup S$. To identify candidate Steiner points, heuristics utilize well-known properties of SMTs. The *angle condition* states that each pair of edges in a SMT meet at an angle of no less than 120°. The *degree condition* states that the degree of each terminal point is at most three and the degree of each Steiner point is exactly three.

In one of the first heuristic approaches for the planar ESTP, Thompson (1973) describes a local search heuristic that utilizes the MST to generate candidate Steiner points. Beginning with $S = \emptyset$, Thompson's algorithm iteratively inserts Steiner points to correct violations of the angle condition. Each insertion involves the identification of a $\triangle uvw$ for which the edges (u, v) and (u, w) violate the angle condition and have the largest dot product of all such violating pairs of edges. Thompson's algorithm executes an insertion by removing the edges (u, v) and (u, w), inserting a Steiner point s at its optimal location relative to the three neighboring points, and adding edges (u, s), (v, s), and (w, s). After executing up to $n - 2$ insertions (an SMT on n terminal points has at most $n-2$ Steiner points (Gilbert and Pollak, 1968)), the positions of the Steiner points are iteratively optimized (treating neighboring points as fixed) until improvement drops below a specified threshold; this step can be viewed as an early relatively minimal tree (RMT) algorithm. Dreyer and Overton (1998) present an extension of Thompson (1973) which capitalizes on improved RMT algorithms by placing the inserted Steiner points on top of one of

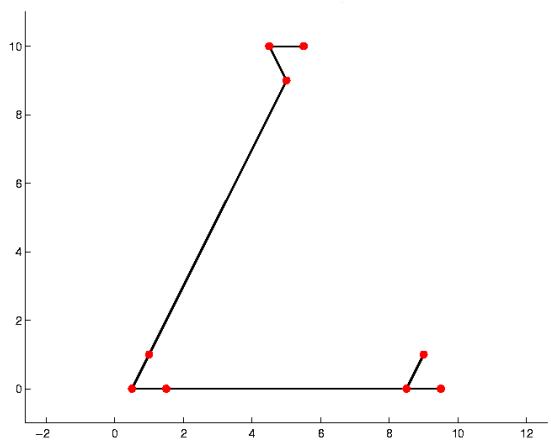


Figure 3.1: The MST for the nine-point instance, which provides the initial solution for Steiner insertion heuristics.

the terminal points and determining the optimal location of all Steiner points in a single call to the interior point method of Andersen and Andersen (1996).

The simple edge insertion heuristics of Thompson (1973) or Dreyer and Overton (1998) essentially perform "local Steinerizations" and are generally oblivious to any global structure. This approach fairs moderately well on random instances, but can suffer severe shortcomings on clustered instances. Figure 3.1 illustrates the MST for a nine-point triangle instance in which these edge insertion heuristics miss the optimal solution. The edge insertion heuristic only positions the Steiner points in the three smaller triangles as in Figure 3.2, but is unable to detect the Steiner point needed in the middle for the optimal solution as shown in Figure 3.3.

Chang (1972) develops a *generalized Steiner insertion* heuristic which begins with the MST and constructs a sub-optimal FST in $n-2$ iterations. Each iteration involves the insertion of a Steiner point, its connection to three terminals (creating two cycles), and the removal of two edges to eliminate the cycles. With each iteration, partial FSTs on subsets of terminal points are enlarged. Finally, the locations of all Steiner points are iteratively optimized as in Thompson (1973).

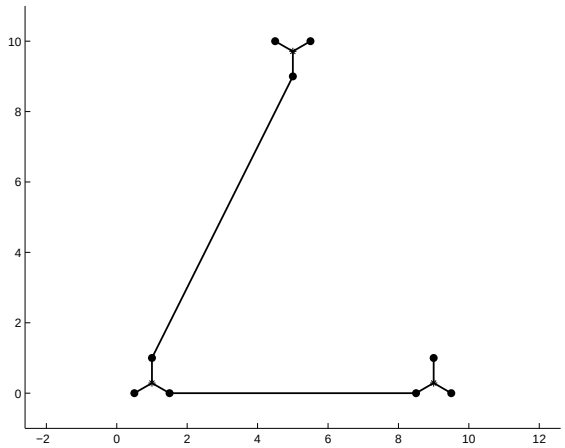


Figure 3.2: Edge insertion achieves a sub-optimal solution to the nine-point instance.

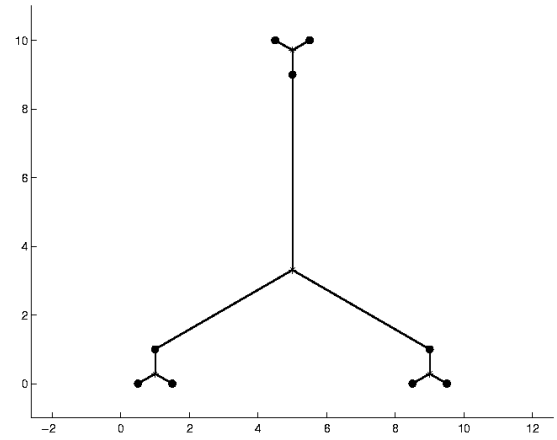


Figure 3.3: The optimal solution for the nine-point instance contains a centrally-located Steiner point.

Beasley and Goffinet (1994) present a more elaborate algorithm which generates candidate Steiner points for the planar ESTP using computational geometry, specifically, the Delaunay triangulation (see Section 2.2). The Delaunay triangulation possesses important properties that explain its use as a mechanism for generating candidate Steiner points in ESTP heuristics. First, the Delaunay triangulation generates simplices such that the smallest angle in any simplex is maximized, i.e, the simplices are approximately equilateral. This is a desirable trait as the maximum length reduction resulting from a Steiner point insertion in the plane occurs in an equilateral triangle. A second property is that the edges of the Delaunay triangulation of a set of points, Z, contain the MST of Z (De Berg et al., 2008). Thus, an algorithm considering Steiner point insertion in Delaunay simplices is more general than an insertion mechanism targeting non-degenerate simplices which have two edges in the MST, as in Smith et al. (1981).

Additionally, the Delaunay triangulation encodes information about the global structure of the set of points. As Zachariasen (1999) notes, connection via Steiner point is largely a local property. As Figures 3.1, 3.2, and 3.3 illustrate, focusing Steiner insertion only for adjacent edges in the MST can be myopic. To avoid this myopic behavior, a type of *generalized* Steiner insertion is required (Chung and Graham, 1978). From the $\binom{n}{d+1}$ sets of simplices for potential Steiner point insertion (in $\Re^d$), the Delaunay triangulation narrows down the candidates for insertion to those corresponding to the sets of vertices that are "close" in the sense that they share a face in the Voronoi diagram. Thus, the Delaunay triangulation provides an intuitive mechanism to consider broader set of candidate Steiner points than the edge insertion approach of Thompson (1973) which considers only those Delaunay simplices containing two MST edges which meet at less than 120°. In Figure 3.4, an insertion in the center Delaunay triangle is sufficient for finding the global solution, as the other three Steiner points can be found via an edge insertion algorithm.

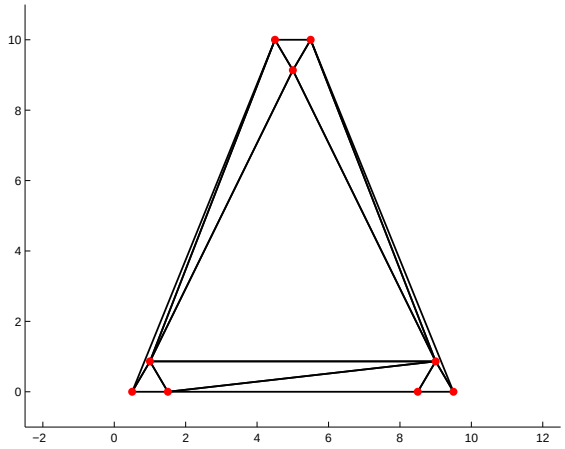


Figure 3.4: The Delaunay triangulation for the nine-point instance identifies the central Steiner point location requisite for the SMT.

To generate a neighbor solution in their heuristic approach, Beasley and Goffinet (1994) iterate between an expansion phase and a reduction phase. The expansion phase augments S, the set of Steiner points, (initially $S = \emptyset$) by placing a Steiner point in each Delaunay triangle of $DT(X \cup S)$ containing no angle greater than or equal to 120°. The subsequent reduction phase forms the MST of $X \cup S$ and applies the degree condition to reduce the number of potential Steiner vertices. This reduction step, referred to as a "cleanup" operation, results in a tree with a *Steiner topology*, i.e., the degree of each terminal point is at most three and the degree of each Steiner point is exactly three. Following the reduction phase, a re-expansion akin to the edge insertion algorithm of Thompson (1973) is performed to correct any violation of the angle condition. To avoid becoming easily trapped in a local minima, Beasley and Goffinet (1994) base the acceptance or rejection of the neighbor solution upon the simulated annealing metaheuristic (Laarhoven and Aarts, 1987; Ingber, 1993)which allows worse solutions (in terms of tree length) to be accepted to enable the discovery of better solutions at later iterations.

3.2.2.2 Concatenation heuristics

Motivated by the observation that the SMT can be viewed of as a concatenation of FSTs on subsets of terminal points (Gilbert and Pollak, 1968), concatenation heuristics attempt to find quality FSTs on subsets of terminals (*partial FSTs*) and concatenate them into the shortest possible tree.

Smith et al. (1981) present a FST concatenation algorithm based on the Delaunay triangulation and MST. In the first phase, the algorithm decomposes the instance into Delaunay triangles and generates partial FSTs by considering Delaunay triangles in which a Steiner point insertion appears promising. All partial FSTs on three terminals are added to the priority queue in the order of increasing value of $\ell(SMT)/\ell(MST)$ for the SMT and MST on the three corresponding points. The next step considers all adjacent pairs of Delaunay triangles which have three combined edges in the MST, and performs a Steiner point insertion to create a FST on four terminals for each set of three such edges. These partial FSTs on four terminals are added to the priority queue in increasing value of $\ell(SMT)/\ell(MST)$. In the second phase, the algorithm greedily concatenates the partial FSTs in the queue with priority based on their reduction in length over the MST. The algorithm of Smith et al. (1981) runs in $\mathcal{O}(n \log n)$ time and generates solutions of equal quality to the $\mathcal{O}(n^4)$ run-time algorithm of Smith and Liebman (1979).

In a similar fashion, Zachariasen (1999) constructs a restricted candidate list, F, of FSTs on subsets of terminal points and then creates a sub-optimal SMT by concatenating elements of this list. Zachariasen (1999) utilizes the *Gabriel graph* of the set of terminal points to select candidate FSTs. The Gabriel graph (GG) is a subset of the Delaunay triangulation (DT) that is found by removing edges between vertices if, in the Voronoi diagram, the edge between them does not intersect the Voronoi face separating them. A known result is that MST $\subset$ GG $\subset$ DT (De Berg et al., 2008).

In order to form the restricted candidate list, F, Zachariasen (1999) considers all connected components of the Gabriel graph up to degree K ($K \leq 5$). If the set of terminal points is in "Steiner configuration" (that is, all terminals lie on the convex hull), Zachariasen forms all FSTs on that subset and add the shortest to F. Zachariasen shows that the number of FSTs for points in Steiner configuration is drastically reduced from the number of FSTs on terminals in general position. In addition, Zachariasen does not consider those FSTs which violate the bottleneck distance (see Section 2.1).

The partial FST concatenation phase of Zachariasen (1999) removes the geometric problem of locating Steiner points and so converts the ESTP into a purely combinatorial problem. The solution spaces consists of all 0-1 vectors of size m', where m' is the number of full topologies on three or more terminals, $S = \{0, 1\}^{m'}$. Zachariasen tests three neighbor generation schemes: a simple flip neighborhood (N_F), a global insertion neighborhood (N_I), and a local insertion neighborhood (N_L). In N_F, a neighbor is generated by removing a partial FST from the incumbent solution, inserting another (previously non-included) partial FST, and subsequently inserting appropriate edges of the MST to regain the tree structure. To create a neighbor solution from N_I, an empty solution is originated with a selected candidate partial FST and compatible FSTs of the incumbent solution are inserted so long as no cycles are created. In this way, a new solution is created by adding a new element and "pushing out" old elements. The N_L neighbor creation scheme consists of inserting partial FSTs in local time by removing all currently conflicting elements and reconnecting with the minimal spanning tree. The first two neighborhoods N_F and N_L possess the attractive feature of being *strongly connected*: any solution is reachable from any other by a sequence of neighbor generations. Each of the neighborhoods is tested with 3 local search methods: repeated descent (RD), simulated annealing (SA) (Laarhoven and Aarts, 1987; Ingber, 1993), and

tabu search (TS) (Glover, 1990). The N_I neighborhood outperforms both the N_L and N_F neighborhoods; under this neighborhood Zachariasen (1999) obtains solutions of better quality than Beasley and Goffinet (1994) across the RD, SA, and TS metaheuristic frameworks (RD and SA significantly outperform TS).

Zachariasen and Winter (1999) present a class of $\mathcal{O}(n \log n)$ greedy concatenation heuristics which identify candidate SMTs on subsets of k terminal points ($k = 3, 4, 5$) using subgraphs of the Delaunay triangulation and the MST. Zachariasen and Winter (1999) select partial FSTs based on structures from computational geometry including the relative neighborhood graph, Gabriel graph, and Delaunay triangulation. Zachariasen and Winter (1999) sort the selected partial FSTs according to their reduction over the MST on the given subset of terminal points as in Smith et al. (1981). Zachariasen and Winter (1999) test a greedy concatenation neighbor generation as in Smith et al. (1981) and the global insertion neighbor generation as in Zachariasen (1999). The values of solutions exceed those found by the previous $\mathcal{O}(n \log n)$ algorithm of Smith et al. (1981).

The methods of Zachariasen (1999) and Zachariasen and Winter (1999) possess several attractive features. The Steiner point insertion heuristics described in Section 3.2.2.1 repeatedly locate Steiner points in order to evaluate the length of the current solution. The algorithms of Zachariasen (1999) and Zachariasen and Winter (1999) avoid this problem by locating Steiner points only once for each partial FST, which are kept small. After locating Steiner points, the length of the partial FST is stored along with the tree. In determining the cost of a candidate tree, it only remains to sum the stored lengths of its partial FSTs. This scheme also has the advantage of avoiding cleanup operations by ensuring every resulting tree possesses a Steiner topology.

The difficulty faced by concatenation heuristics is the construction of F, the candidate list of FSTs. A possible drawback of this approach is that the candidate

list, F, established by an algorithm may not contain all FSTs needed to create the SMT. An omitted partial FST can never appear in the generated solution. On randomized instances with $k = 5$, the algorithm of Zachariasen and Winter (1999) typically omits less than 0.12 partial FSTs which are necessary to construct the SMT. By restricting the size of partial FSTs considered to 5 or fewer terminals, the algorithm could perform poorly on non-degenerate ESTP instances with greater than five terminals.

3.2.2.3 Other heuristic methods

We detail several heuristics which are neither Steiner point insertion heuristics or concatenation methods; rather, these heuristics consider only FSTs. Smith (1992) proves that every degenerate Steiner topology is equivalent to one (or more) FSTs with zero length edges. Such a solution can be obtained by concatenating partial FSTs as in Section 3.2.2.2 or by considering only FSTs (realizing there will be zero length edges after calling a RMT algorithm). Heuristics which consider FSTs need only store the topology of the FST and so convert the ESTP into a purely combinatorial problem. An initial solution is a full Steiner topology, and a neighbor is generated by modifying the current FST into another FST. Smith (1992) shows that all FSTs on n terminals are represented by the set of vectors of length $n - 3$ where the i^{th} entry, x_i, is a natural number $1 \leq x_i \leq 2 * i - 1$.

Lundy (1985) places a full Steiner topology perturbation approach into a simulated annealing framework. The initial solution is a random full Steiner topology with $n - 2$ randomly located Steiner points. The neighbor generation is a topology perturbation followed by a Steiner point relocation scheme which can be viewed as an early RMT algorithm. The topology perturbation consists of: 1) removing an "external branch" (i.e., one terminal point and one Steiner point) and repairing the tree to create a full topology on $n - 1$ terminal points (and one lone terminal), and 2) selecting a new edge to which to connect the terminal point via a Steiner point.

A FST on n terminals has $2n-3$ edges, so there are $2n-4$ potential neighbors for each choice of external branch (of which there are n choices). This quadratic-sized neighborhood also has the attractive feature of being strongly connected.

Dreyer and Overton (1998) present a constructive "incremental optimization" (IO) heuristic, which is a greedy variant of the exact branch-and-bound algorithm of Xue et al. (1999) and Smith (1992). The algorithm begins with a FST on three of the terminal points, referred to as level $k = 1$. At each level k, an incumbent solution is a partial FST with k Steiner points and $k+2$ terminals. To generate a candidate solution for level $k+1$, one of the $(n-k-2)$ unselected terminal points is chosen and connected via Steiner point to an existing edge, creating a partial FST with $k+1$ Steiner points on $k+3$ terminals. For a selected terminal point, $2k+1$ candidate solutions are generated for level $k+1$ since there are $2k+1$ edges in the incumbent solution at level k. An interior point algorithm (Andersen and Andersen, 1996) for the minimum-sum-of-norms problem optimizes the location of Steiner points to determine the length of the candidate partial FSTs created. The shortest of the created partial FSTs becomes the new incumbent, and the process repeats a total of $n-2$ times to construct a FST on the n terminals. The algorithm produces solutions of better quality than an edge insertion heuristic (Dreyer and Overton, 1998), but requires significantly more computational time.

3.2.3 Heuristics in $\mathcal{R}^d$

Relative to the planar ESTP, the literature on heuristics for the ESTP in $\Re^d$ for $d > 2$ is limited. The planar FST concatenation heuristics, for example, heavily utilize planar geometry to vastly reduce the size of the candidate list F and these conditions do not easily generalize to higher dimensions. The algorithms of Zachariasen (1999) and Zachariasen and Winter (1999) utilize fathoming criteria based on "equilateral points" to restrict the size of the list F, thus do not extend to higher dimensions.

Ravada and Sherman (1994) apply a partitioning technique for the problem in $\Re^d$, $d = 2, 3$. A parameter t governs the size of the subproblems created in the partition, and each of these subproblems is solved by the exact algorithm of Smith (1992). Smith et al. (1995) utilize the Delaunay triangulation and MST to decompose specially-structured sausage-shaped instances into smaller subproblems that are solved with an exact algorithm. Smith et al. (1995) then link these partial solutions to form a sub-optimal SMT. Toppur and Smith (2005) extend the heuristic to random instances in $\Re^3$ using decomposition by Delaunay tetrahedrons.

3.3 Exact algorithms

We review background information for exact ESTP algorithms and contrast the planar algorithm and the algorithm for higher dimensions.

3.3.1 Planar algorithms

Exact algorithms for the planar ESTP use "equilateral points" to accelerate the search for the SMT, but this mechanism also prevents these algorithms from generalizing to higher dimensions. Given two points x_i and x_j, there are two equilateral points a and b: the vertices of equilateral triangles $\triangle a x_i x_j$ and $\triangle b x_i x_j$. Utilizing equilateral points, Melzak (1961) proposed the first finite-time algorithm for the ESTP in the plane. The proposed algorithm i) enumerates all FSTs on the n terminals and ii) optimizes the locations of Steiner points for each FST using equilateral points. Optimizing the location Steiner points via equilateral points is done with a reduction operation, where two terminals are replaced by an equilateral point (which is then treated as a terminal, and the reduction is repeated). When the reduction is completely, the equilateral points are iterative replaced by the terminals they encode. For each pair of terminals, there are two choices for equilateral point so the algorithm for optimizing the location of Steiner points runs in $\mathcal{O}(2^k)$ time. Hwang (1986) improves the algorithm by eliminating one of the equilateral points, resulting in an $\mathcal{O}(k)$ time algorithm.

Winter (1985) develops an exact approach for the planar ESTP by enumerating equilateral points *rather* than FSTs. The GeoSteiner algorithm (Warme et al., 2001) i) generates all FSTs on subsets of terminals which may appear in the SMT and ii) exhaustively concatenates the list of partial FSTs to create the SMT. Enumeration via equilateral points has the attractive feature that many partial FSTs do not survive the generation process, and further geometric conditions are checked for those partial FSTs which do survive. Advances in the fathoming criteria have lead to increases in the size of ESTP instances which can be solved to optimality by the GeoSteiner algorithm (Warme et al., 2001), which can optimally solve planar ESTP instances with up to 1000 terminal points.

The finite cardinality of the set of equilateral points is a critical feature used by the algorithms of Melzak (1961), Winter (1985), and Winter and Zachariasen (1997). In higher dimensions, equilateral points become equilateral-arcs. Rather than checking one or two equilateral points for each pair of terminals, an entire arc of equilateral points would need to be checked. It is then impossible to enumerate FSTs via the now uncountable set of equilateral points.

3.3.2 Smith's algorithm in $\mathcal{R}^d$

The algorithm of Smith (1992) enumerates only full topologies on all n terminal points, and it does so by "growing" a partial FST. Let a set of n terminal points in $\mathbb{R}^d$ be given. Smith's enumeration begins with a full Steiner topology on three of the terminal points (recall there is only one such FST). At each subsequent level of the branch-and-bound tree, a terminal point is "sprouted" or "merged" from the existing tree by connecting it via a Steiner point to an existing edge (see Figure 3.5). In the first level ($k = 1$), there will be $2k+1 = 3$ children of this parent node. Smith demonstrates that performing this merge operation $n - 2$ times generates all full Steiner topologies (FST) on the n terminal points. He also shows that the merge operation can not *decrease* the length of the tree. Thus, if an existing full Steiner

tree on a subset of terminal points is longer than a known Steiner tree on the set of n terminals, this partial FST and its descendants may be removed from consideration (fathomed).

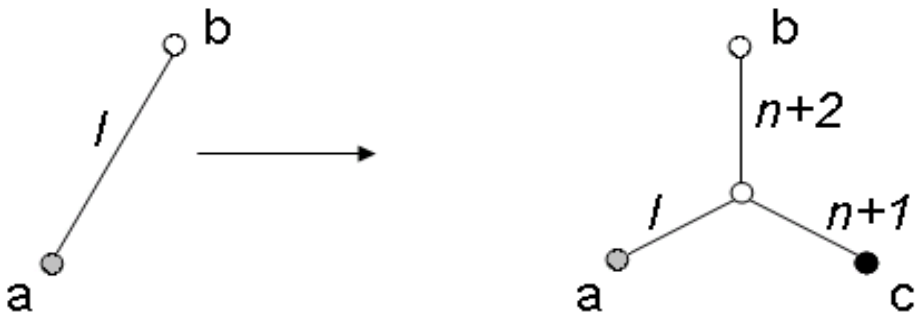


Figure 3.5: The "merge" operation used in Smith's enumeration scheme.

The state-of-the-art exact algorithm for the ESTP in $\Re^d$ for $d > 2$ is the Smith+ algorithm (Fampa and Anstreicher, 2008) which enhances the algorithm of Smith (1992) by implementing second-order cone programming (see Section 2.3) to locate the Steiner points and "strong branching" to accelerate the fathoming process in the branch-and-bound enumeration scheme. Strong branching merges terminal points so that that the FST grows in length as quickly as possibly, allowing fathoming higher in the branch-and-bound tree. At each level of the branch-and-bound tree, the Smith+ algorithm solves a sub-problem to choose which terminal to merge next. It chooses the terminal so that the most children possible will be fathomed at the next level. The Smith+ algorithm is capable of optimally solving instances with up to 16 terminal points in $\Re^d$ for $2 \leq d \leq 5$.

The algorithms in $\Re^d$, $d > 2$ have several drawbacks. A major deficiency of Smith's enumeration scheme is that the length-based fathoming criteria is quite weak and can not be expected to remove nodes from consideration until deep in the branch-and-bound tree. Neither Smith's original algorithm nor the Smith+

algorithm use initial upper bounds to fathom nodes; the only bounds used are those achieved by the algorithm itself. Another interesting feature is that the algorithms make *no* use of geometry whatsoever, a feature we incorporate in Chapter 5.

CHAPTER 4
DELAUNAY TRIANGULATION HEURISTIC

This section describes the heuristic algorithm for the ESTP in $\Re^d$, $2 \leq d \leq 5$. We utilize the Delaunay triangulation to generate candidate Steiner points for insertion, the MST to identify Steiner points to be removed, second-order cone programming to optimize the location of the remaining Steiner points, and an edge insertion similar to Dreyer and Overton (1998) as an attempt to correct any violations of the angle condition. We govern the neighbor generation procedure with a first-improvement local search framework, as outlined in Algorithm 4.1.

4.1 Description of the heuristic

Recall that X denotes the set of terminal points in Euclidean space, and S is a set of Steiner points. We initialize $S = \emptyset$, and then construct the Delaunay triangulation of $X \cup S$. In each Delaunay simplex, we insert a Steiner point at the simplex centroid with probability p; where we randomly select the value of p from the range $[l, u]$ for $0 \leq l \leq u \leq 1$. We generate a candidate tree, T', by forming the MST on X and the newly augmented set of Steiner points.

After forming the MST on $X \cup S'$, T' may have a non-Steiner topology and warrant cleanup operations. Therefore, via Algorithm 4.2, we iteratively remove all Steiner points of degree one or two (and appropriately repair the tree) until there are no Steiner points of degree less than three. Then we (locally) optimize the location of each Steiner point of degree three; we determine these optimal locations analytically in the plane and computationally via SeDuMi's second-order cone solver (Sturm and Polik, 2006) for $d > 2$. After the removal of the low-degree Steiner points and the subsequent repair, the tree may contain ill-suited Steiner connections between distant terminal points. We form the MST on the terminal points and the current set of Steiner points which serves an intermediate check on the validity of the

topology. This reformation of the MST will eliminate unnecessarily long edges, but may re-introduce Steiner points of degree one or two, requiring removal operations to be repeated.

Before exiting the cleanup subroutine, we check the tree for the existence of Steiner point to Steiner point edges. If such edges exist, then the local optimization of degree three Steiner points in the previous steps may not be the globally optimally configuration for the tree. In such a case, we invoke the second-order cone solver of SeDuMi (Sturm and Polik, 2006) to optimize the locations of all Steiner points (global optimization). For details regarding formulating the problem of locating the Steiner points for a given full Steiner topology as a conic optimization problem, see Section 2.3. The optimization of the Steiner point locations can result in Steiner points coalescing with neighboring Steiner or terminal points, and in such a case we remove degenerate Steiner points and appropriately reattach the tree.

After completing the cleanup operations of Algorithm 4.2, we perform an edge insertion as outlined in Algorithm 4.3 as a final adjustment to the candidate tree. The steps of this edge insertion procedure are the same as in Dreyer and Overton (1998), but differ in the edge selection criteria. We select violating edges based on largest dot product, while Dreyer and Overton (1998) prioritize via smallest violating angle. See Section 3.2.2.1 for a detailed description of Dreyer and Overton (1998). Following the edge insertion, we compute the length of the candidate tree, $\ell(T')$, and if it is within χ percent of the current tree's length, we execute Algorithm 4.4 to ensure the candidate tree possesses a Steiner topology. An iteration concludes with comparing the lengths of the candidate tree and current tree to execute the first-improvement local search. If the current solution is not improved in *trial limit* iterations, the algorithm terminates.

Algorithm 4.1 Local Search

Input: Set of terminal points, X, in $\Re^d$
Output: A tree, T, with Steiner topology on $X \cup S$, where S is a set of Steiner points.
Initialization:
Set $counter = 1$ and $S = \emptyset$.
Set $T = MST(X)$.
while $counter \leq trial\ limit$ **do**
Construct $DT(X \cup S)$.
Randomly generate the insertion probability, p, from the range $[l, u]$.
for each Delaunay simplex **do**
With probability p, insert a Steiner point s and let $S' = S \cup s$.
end for
Set $T' = MST(X \cup S')$.
$T' \leftarrow Clean(T')$ via Algorithm 4.2.
$T' \leftarrow EdgeInsertion(T')$ via Algorithm 4.3.
if $\ell(T') - \ell(T) < \chi\ell(T)$ **then**
$T' \leftarrow Recover(T')$ via Algorithm 4.4.
end if
if $\ell(T') - \ell(T) < 0$ **then**
Set $S = S'$, $T = T'$, and $counter = 1$.
else
$counter \leftarrow counter + 1$.
end if
end while

Algorithm 4.2 Cleanup Procedure

Input: A tree, T', on a union of terminal points and Steiner points, $X \cup S'$.
Output: A tree, T'', on a union of terminal points and Steiner points, $X \cup S''$.
Initialization:
Set $S'' = S'$ and $T'' = T'$.
repeat
  for each Steiner point $s \in S''$ **do**
    if $degree(s) = 1$ **then**
      Delete the edge adjacent to s from T''.
      $S'' \leftarrow S'' - s$.
    else
      if $degree(s) = 2$ **then**
        Delete the two edges, (s, x) and (s, y), adjacent to s from T''.
        Insert a new edge, (x, y), into T''.
        $S'' \leftarrow S'' - s$.
      end if
    end if
  end for
  for each Steiner point $s \in S''$ **do**
    if $degree(s) = 3$ **then**
      Optimize the location of s.
    end if
  end for
  Set $T'' = \text{MST}(X \cup S'')$
until each Steiner point $s \in S''$ has degree ≥ 3
if there exists at least one edge between Steiner points in T'' **then**
  Optimize location of the Steiner points in S'' via SeDuMi (Sturm and Polik, 2006).
end if

Algorithm 4.3 Edge Insertion (Dreyer and Overton, 1998)

Input: A tree, T, on a union of terminal points and Steiner points, $X \cup S$.
Output: A tree, T', on a union of terminal points and Steiner points, $X \cup S'$.
Initialization:
Let E be the set of edges in T.
Set $T' = T$, $S' = S$, and *edge list* $= \emptyset$.
for $i = 1, \ldots, |X \cup S|$ **do**
  for j such that $(x_i, x_j) \in E$ **do**
    for l such that $(x_j, x_l) \in E$ **do**
      if (x_j, x_l) meets (x_i, x_j) at angle less than 120° **then**
        edge list $\leftarrow$ *edge list* $\cup (x_j, x_l)$.
      end if
    end for
    if *edge list* $\neq \emptyset$ **then**
      From *edge list*, select (x_j, x_k) which maximizes dot product with (x_i, x_j).
      Let $s = x_j$, let $S' = S \cup s$.
      Remove edges (x_i, x_j) and (x_j, x_k) from T'.
      Add edges (x_i, s), (x_j, s), and (x_k, s) to T'.
      Set *edge list*$= \emptyset$.
    end if
  end for
end for
Optimize location of the Steiner points in S' via SeDuMi (Sturm and Polik, 2006).

Algorithm 4.4 Steiner Topology Recovery

Input: A tree, T, on a union of terminal points and Steiner points, $X \cup S$.
Output: A tree, T', with a Steiner topology on $X \cup S'$.
Initialization:
Let E be the set of edges in T.
Set $T' = T$, $S' = S$, and *edge list* $= \emptyset$.
for $i = 1, \ldots, |X \cup S|$ **do**
  if $degree(x_i) > 3$ **then**
    Set *edge list* $= \emptyset$
    for j such that $(x_i, x_j) \in E$ **do**
      edge list $\leftarrow$ *edge list* $\cup (x_i, x_j)$.
    end for
    for $j = 1, \ldots, degree(x_i) - 3$ **do**
      Select edges, (x_i, x_k) and (x_i, x_l), from *edge list* with largest dot product.
      Let $s = x_i$, let $S' = S \cup s$.
      Remove edges (x_i, x_k) and (x_i, x_l) from T' and from *edge list*.
      Add edges (x_i, s), (x_k, s), and (x_l, s) to T'.
    end for
  end if
end for
Optimize location of the Steiner points in S' via SeDuMi (Sturm and Polik, 2006).

4.2 Algorithm features

The algorithm maintains many features of other ESTP heuristics relying upon the Delaunay triangulation but with several important differences. One key difference is that we probabilistically insert Steiner points into each Delaunay triangle (including the degenerate Delaunay triangles) rather than deterministically inserting Steiner points into all non-degenerate Delaunay triangles (Beasley and Goffinet, 1994) or only inserting Steiner points into Delaunay triangles with two edges in the MST (Smith et al., 1981). The difference in insertion strategy allows our algorithm to achieve different FSTs on subsets of terminal points.

Figures 4.1, 4.2, 4.3, and 4.4 demonstrate how a probabilistic Steiner insertion can quickly achieve FSTs that a deterministic procedure may not. Figure 4.1 illustrates the minimal spanning tree and corresponding Delaunay triangulation for the sixth of the size-10 ESTP instances from the OR-Library (Beasley, 1990). This is an instance for which the approaches of Smith et al. (1981) and Beasley and Goffinet (1994) struggle to achieve a SMT. Our local search heuristic finds the optimal solution in Figure 4.2. The key region is the two lower left-hand Delaunay triangles. Figure 4.3 shows that deterministic insertion in all non-degenerate Delaunay triangles results in a pair of degree-two Steiner points in this lower left-hand region that will be removed after cleanup. Performing six iterations of expansion / reduction phase as prescribed by Beasley and Goffinet (1994) fails to uncover the lower FST existing in the SMT of Figure 4.2. While this empirical observation does not unequivocally guarantee that Beasley and Goffinet (1994) cannot achieve a SMT for this instance (a SMT may be reachable from another solution visited in the simulated annealing run), it reveals insight on the effectiveness of probabilistic Steiner point insertion. In this example, Figure 4.4 shows how probabilistically inserting in the left Delaunay triangle in the lower left-hand region, but not the right Delaunay triangle, quickly results in the key FST being formed in the lower left

corner, eventually leading to the SMT of Figure 4.2.

While Beasley and Goffinet (1994) do not consider randomization in the generation of their candidate solutions, they accept non-improving solutions probabilistically depending on the current temperature of the simulated annealing scheme. That is, Beasley and Goffinet (1994) generate neighbor solutions *deterministically* and accept them *probabilistically*, while we generate neighbor solutions probabilistically and accept them deterministically. That is, the probabilistic mechanism in the acceptance function of simulated annealing has been removed in lieu of a probabilistic neighbor generation procedure. As computational results in §4.3 attest, the probabilistic neighbor generation scheme quickly generates a diverse set of topologies to allow an effective sampling of the search space.

In addition to a difference in insertion criteria, we simplify the method for initially locating the inserted Steiner points. We insert Steiner points via centroid rather than optimally locating the Steiner point relative to its neighbors. This location is more efficient to calculate than the floating point arithmetic operations to analytically locate a Steiner point or to solve a second-order cone program via SeDuMi (Sturm and Polik, 2006). Furthermore, it is unclear how to optimally place a single Steiner point in Delaunay simplices for $d > 2$ as a Delaunay simplex in dimension d is defined by $d + 1$ points which requires $d - 1$ Steiner points to form a FST. The issue is addressed by simply inserting a single point at the simplex centroid and allowing later iterations of the algorithm to insert additional Steiner points in this simplex.

We simplify the post-processing procedures applied to the candidate tree after the trial insertion of Steiner points relative to those of Beasley and Goffinet (1994). Beasley and Goffinet (1994) run an optimization subroutine on Steiner points of degree four in order to determine the best FST on those four points, and delete all Steiner points of degree greater than or equal to five. In contrast, the

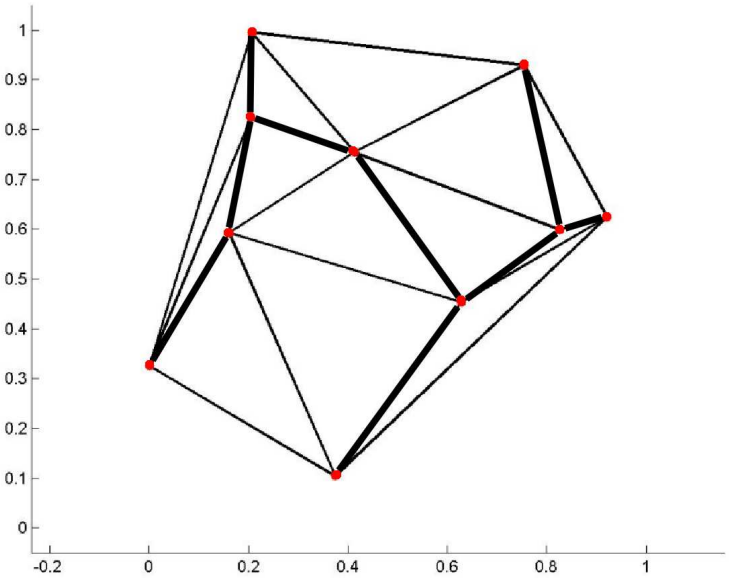


Figure 4.1: As demonstrated by the sixth instance of the size-10 ESTP instances from the OR-Library, the Delaunay triangulation contains the edges of the MST (highlighted in bold).

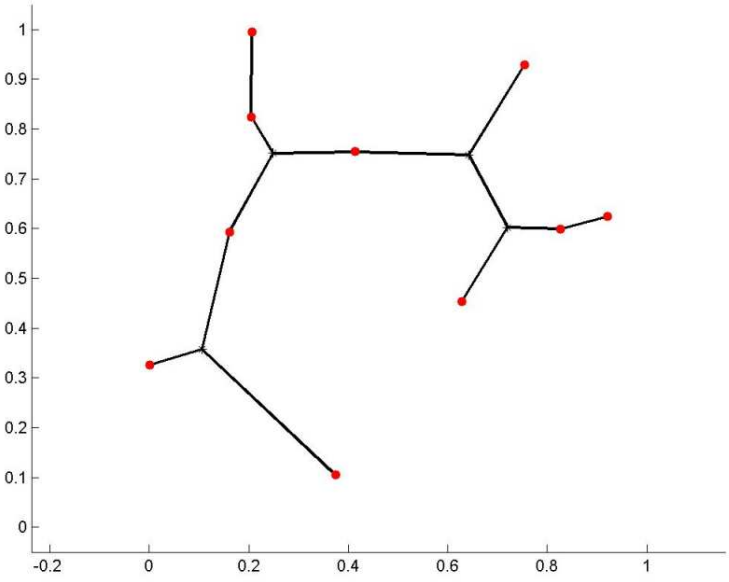


Figure 4.2: A SMT for the sixth instance of the size-10 ESTP instances from the OR-Library contains a FST on three terminal points in the lower left-hand region of the network.

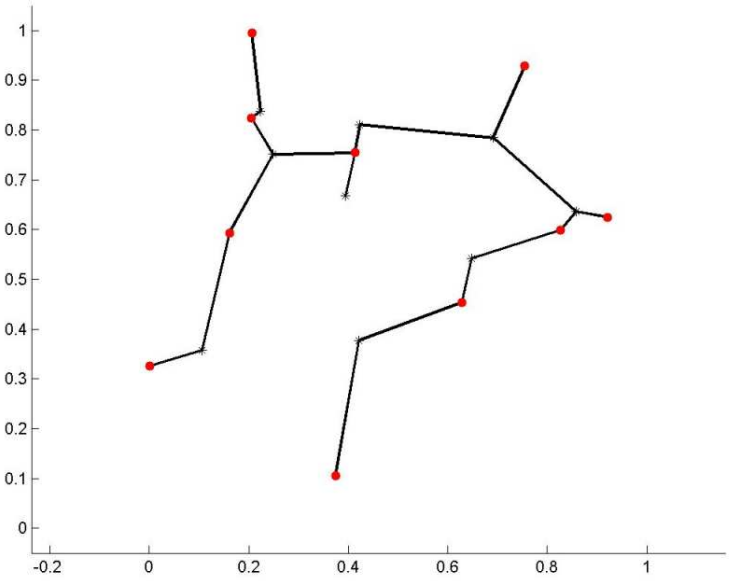


Figure 4.3: Deterministic Steiner point insertion in all nine non-degenerate Delaunay triangles results in a pair of degree-two Steiner points in the lower left-hand region which will be removed by a cleanup operation.

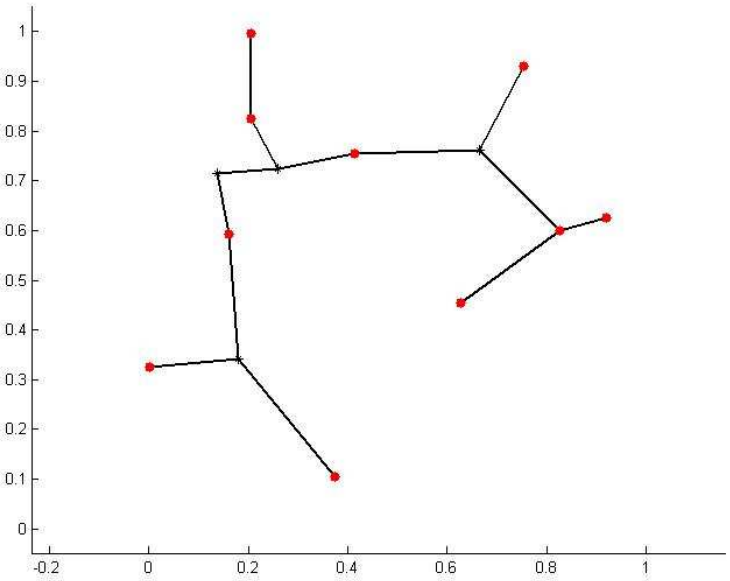


Figure 4.4: Probabilistic Steiner point insertion identifies the lower left FST in the SMT.

cleanup procedure (Algorithm 4.2) addresses only Steiner points of degree less than three and attempts to address the remaining untenable Steiner points by inserting edges via Algorithm 4.3. The applications of Algorithms 4.2 and 4.3 may result in a candidate tree that still lacks a Steiner topology (and thus could possibly be shortened). Rather than attain a Steiner topology for all candidate solutions, we utilize Algorithm 4.4 to directly address points with degree larger than three only in candidate trees within χ percent of the current solution. The computational work, e.g., Table 4.6 in §4.3, suggests that addressing high-degree points via Algorithm 4.4 typically does not result in a large improvement. Thus, setting χ to be a small value will reduce computational effort by only confirming a Steiner topology for candidate trees with a strong potential to improve the current solution.

4.3 Computational results

We implement the algorithm in Matlab and run computational experiments on a dual core 2.4 GHz Pentium processor with 2 GB of RAM and a Windows operating system. We compare the algorithm to Beasley and Goffinet (1994) and Zachariasen (1999) on 195 planar ESTP instances from the OR-Library (Beasley, 1990). We also test the algorithm on 20 random instances in $\Re^d$, $3 \leq d \leq 5$ from Fampa and Anstreicher (2008) for which the exact solution is known. To determine the quality of solutions computed over highly structured sets in $\Re^3$, we test the algorithm on 10 R-sausage instances of varying size as well as 16 sausage-type instances appearing in Smith et al. (1995) and Toppur and Smith (2005). We compute solution quality using percent reduction over the MST. That is, if T is a candidate Steiner tree for a set of points X, define $\sigma = 100 * (\ell(MST) - \ell(T))/\ell(MST)$ as the percent reduction over the MST. The ratio $\rho = \ell(T)/\ell(MST)$ is also commonly used as a measure of quality, and this is adopted for comparison to Smith et al. (1995) and Toppur and Smith (2005). One converts easily between these two measures of quality as $\sigma = 100 * (1 - \rho)$.

Due to its stochastic nature, we run the algorithm twenty times on each instance and report statistics regarding the best solution, average solution, and performance variability. Tables 4.2 and 4.3 present results by aggregating individual instances by problem size. Each entry in Table 4.2 is computed over 15 instances. Each entry of Table 4.3 is calculated over ten instances for $d = 3$ and over five instances for $d = 4$ and 5. Specifically, let σ_{ij}^n denote the percent reduction obtained on run j of instance i of problem size n. For an individual instance i of size n, let the best percent reduction be $\sigma_{i,best}^n = \max\{\sigma_{i1}^n, ..., \sigma_{i,20}^n\}$ and let the average percent reduction be $\bar{\sigma}_{i\cdot}^n = (\sigma_{i1}^n + \ldots + \sigma_{i,20}^n)/20$. For a problem class of m instances of size n, we report the overall best percent reduction as $\sigma_{best}^n = (\sigma_{1,best}^n + \ldots + \sigma_{m,best}^n)/m$ and the overall average percent reduction as $\bar{\sigma}^n = (\bar{\sigma}_{1\cdot}^n + \ldots + \bar{\sigma}_{m\cdot}^n)/m$. The variability of solution quality for the aggregated results is measured by calculating a pooled estimate of the common variance for each of the instances in the problem class. The pooled estimate is given by $\sqrt{SS_E/(19m)}$, where m is the number of different instances within the problem class, and $SS_E = \sum_{i=1}^{m} \sum_{j=1}^{20} (\sigma_{ij} - \bar{\sigma}_{i\cdot})^2$ is the sum of the squares due to error within instances (Montgomery, 2001). We report computation time (in CPU seconds) as the average time per run.

We report the performance for individual instances from Smith et al. (1995) and Toppur and Smith (2005) for the benchmarks in Tables 4.4 and 4.5 in terms of ρ. Specifically, $\rho_{best} = \min\{\rho_1, ..., \rho_{20}\}$ represents the best solution obtained out of twenty runs on a specified instance. Similarly, average solution quality is given by $\bar{\rho} = (\rho_1 + \ldots + \rho_{20})/20$ and performance variability is given by the standard deviation of $\rho_1, \ldots, \rho_{20}$.

To demonstrate the role of the insertion probability, p, we first perform computational experiments in which p is fixed at various values ranging from 0 to 1 and execute Algorithm 4.1 on the planar ESTP instances from the OR-Library (Beasley,

1990). Each entry of Table 4.1 represents $\bar{\sigma}^n$ corresponding to a value of p on instances of size n. As values of p between 0.3 and 0.6 consistently achieve the best average percent reductions, we set $l = 0.3$ and $u = 0.6$ in the remainder of our computational testing.

We perform computational tests to establish a value for *trial limit* to control the loop iterations for the local search. Figure 4.5 illustrates the effectiveness of our algorithm, in terms of average percent reduction, when *trial limit* $= k\sqrt{n}$ for various values of k. In the remainder of the computational experiments, we set $k = 3$, i.e., *trial limit* $= 3\sqrt{n}$. In addition, we set $\chi = 0$ so that we only confirm the Steiner topology of candidate trees that are shorter than the current solution.

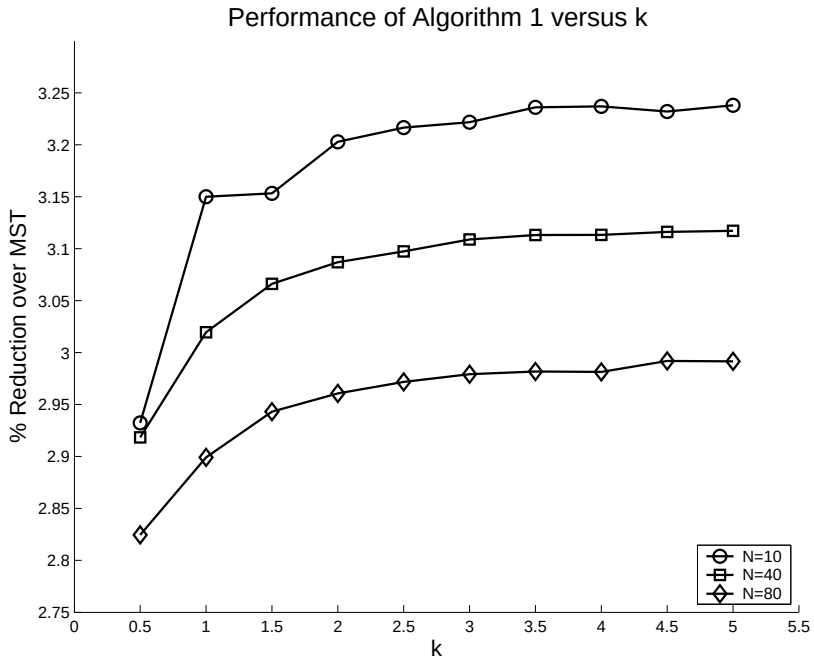


Figure 4.5: For instances of size 10, 40, and 80 from the OR-Library, this graph of average percent reduction versus k, where *trial limit* $= k\sqrt{n}$, demonstrates that a value of $k = 3$ results in an iteration count that provides an acceptable tradeoff between solution quality and solution time.

Table 4.1: Insertion probabilities between 0.3 and 0.6 consistently achieve the largest average percent reduction for the OR-Library ESTP instances.

	Insertion Probability, p											
n	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	OPT
10	2.87	3.04	3.16	3.20	3.20	3.21	3.22	3.20	3.16	3.12	2.95	3.25
20	2.82	3.03	3.09	3.12	3.10	3.12	3.10	3.10	3.06	3.01	2.41	3.16
30	2.80	2.98	3.00	3.02	3.01	3.01	3.03	3.00	2.97	2.91	2.69	3.07
40	2.79	3.06	3.08	3.08	3.08	3.07	3.07	3.06	3.01	2.99	2.85	3.14
50	2.73	2.95	2.98	2.99	2.99	2.99	2.96	2.95	2.89	2.86	2.65	3.03
60	2.74	3.19	3.20	3.21	3.21	3.20	3.20	3.17	3.15	3.06	2.78	3.27
70	2.62	3.02	3.05	3.06	3.06	3.06	3.05	3.04	3.01	2.97	2.74	3.11
80	2.63	2.96	2.96	2.97	2.96	2.95	2.94	2.92	2.88	2.82	2.61	3.04
90	2.79	3.01	3.02	3.03	3.02	3.00	3.01	2.99	2.94	2.91	2.57	3.12
100	2.80	3.17	3.18	3.18	3.17	3.16	3.15	3.11	3.08	3.01	2.59	3.27

Table 4.2 compares the performance of our randomized Delaunay triangulation heuristic to two leading heuristics for the planar ESTP instances from the OR-Library (Beasley, 1990). The problem size ranges from 10 terminal points to 100 terminal points, with 15 instances at each problem size. As Table 4.2 shows, the heuristic typically dominates the simulated annealing approach of Beasley and Goffinet (1994). Based on average performance, our approach does not achieve as large of percent reduction as the FST-local search approach of Zachariasen (1999), which utilizes planar-specific rules to generate a good, short list of FST candidates for concatenation. Our general approach does not take advantage of problem dimension, but we note that the best percent reductions are competitive with the average results of Zachariasen (1999) and that our performance variability is small, suggesting that these best percent reductions are not sampling anomalies (Zachariasen (1999) does not report results analogous to our σ^n_{best} calculations). Comparing computation time across platforms is precarious, but taking into account machine speeds, the running time of our approach is longer than either Beasley and Goffinet (1994) or Zachariasen (1999). We attribute much of this discrepancy to the inefficiency of our Matlab implementation relative to the FORTRAN and C++ implementations of Beasley and Goffinet (1994) and Zachariasen (1999), respectively. We note that a comparison of our termination criteria and the termination criteria of the FST-local search in Zachariasen (1999), which limits the number of descents to $10\sqrt{n}$ and the maximum total iterations to $50\sqrt{n}$, suggests that our termination criteria results in significantly fewer iterations.

We demonstrate our heuristic's robustness by considering the randomized instances from Fampa and Anstreicher (2008) in $\Re^d$ for $3 \leq d \leq 5$. Fampa and Anstreicher (2008) use the Smith+ algorithm to determine optimal solutions for ten instances in $\Re^3$ and five instances each in $\Re^4$ and $\Re^5$. Fampa and Anstreicher (2008) create these instances by randomly generating ten points in the $[0, 10]^d$ cube.

Table 4.2: Our heuristic is competitive with other approaches on the planar OR-Library instances.

	Beasley and Goffinet (1994)		Zachariasen and Winter (1999)		Randomized Delaunay Triangulation			
n	Avg. % Reduction	CPU (min)	Avg. % Reduction	CPU (sec)	σ^n_{best}	$\bar{\sigma}^n \pm \sqrt{SS_E/(9m)}$	CPU (sec)	Opt. % Reduction
10	3.22	0.057	3.23	0.2	3.25	3.22 ± 0.151	0.7	3.25
20	3.12	0.276	3.15	0.9	3.15	3.13 ± 0.058	2.0	3.16
30	2.95	0.609	3.06	2.2	3.07	3.03 ± 0.068	3.9	3.07
40	2.97	1.288	3.12	4.4	3.14	3.11 ± 0.051	6.8	3.14
50	2.92	1.929	3.03	7.2	3.03	3.00 ± 0.049	9.0	3.03
60	3.18	2.595	3.27	9.6	3.27	3.23 ± 0.058	12.9	3.27
70	2.95	3.507	3.11	13.2	3.11	3.08 ± 0.050	17.5	3.11
80	2.92	6.506	3.03	19.2	3.03	2.98 ± 0.048	22.2	3.04
90	2.95	7.800	3.11	23.1	3.11	3.04 ± 0.062	29.3	3.12
100	3.07	8.140	3.25	34.5	3.26	3.19 ± 0.061	40.2	3.27

As Table 4.3 illustrates, our approach produces optimal or near-optimal solutions. For $\Re^d$ for $3 \leq d \leq 5$, our average solutions are within 1.4 percent, 3.4 percent, and 3.3 percent of optimality, respectively. Our best solutions are optimal for all instances. In addition to the best and average percent reduction calculations, we also provide the number of times our algorithm finds the optimal solution in Table 4.3 to demonstrate that our approach consistently achieves the optimal solution. While exact comparisons of computation time are difficult due to varying machine speeds and software implementations (Fampa and Anstreicher (2008) implement the Smith+ algorithm in C, while we implement our approach in MATLAB), we can safely claim that our approach achieves comparable results in the fraction of the time.

We also test the heuristic on 10 $\Re$-sausage and 16 sausage-type networks in $\Re^3$ appearing in Smith et al. (1995) and Toppur and Smith (2005). The $\Re$-sausage is a chain of regular tetrahedra joined together face-to-face, and the sausage-type network instances are concatenations of small sausages meeting at a common face. Toppur and Smith (2005) establish the optimal Steiner topology for the specially-structured $\Re$-sausage instances in Table 4.4, and provide the optimal solutions. As Table 4.4 shows, our heuristic consistently achieves optimal solutions on the $\Re$-sausage instances. Although these instances are specially structured to accommodate the sausage heuristic of Toppur and Smith (2005), our heuristic approach is very competitive.

Table 4.5 contains multiple types of the sausage-type network instances in $\Re^3$. The naming convention for these instances is listed as an acronym. The first letter is either "S" or "L" referring to short or long chains. The second letter is either "S" or "A" referring to symmetric or asymmetric configurations. The final character in the instance name refers to the cardinality of the junctions at which sausage chains intersect ("M" stands for "multiway" junction in which more than

four chains intersect). We refer to Smith et al. (1995) for more details on these constructions.

Table 4.3: Our heuristic achieves results comparable with the exact approach of Fampa and Anstreicher with a fraction of the computational effort.

		Smith+ Algorithm		Randomized Delaunay Triangulation			
		Opt. %	CPU			Optimal	CPU
d	n	Reduction	(sec)	σ^n_{best}	$\bar{\sigma}^n \pm \sqrt{SS_E}$	Runs	(sec)
3	10	5.584	754	5.584	5.506 ± 0.178	171/200	3.9
4	10	8.301	5736	8.301	8.019 ± 0.366	57/100	5.4
5	10	8.229	4681	8.229	7.957 ± 0.371	47/100	6.8

Table 4.4: Our heuristic consistently obtains optimal or near-optimal solutions on the $\Re$-sausage instances in $\Re^3$.

	Toppur and Smith (2005)	Randomized Delaunay Triangulation		
n	$\rho^\star$	ρ_{best}	$\bar{\rho} \pm$ st. dev.	CPU (min)
6	0.80807	0.80807	0.80807 ± 0	0.03
7	0.80286	0.80286	0.80286 ± 0	0.04
8	0.80090	0.80090	0.80090 ± 0	0.05
9	0.79870	0.79870	0.79888 ± 0.0008	0.07
10	0.79701	0.79701	0.79759 ± 0.0014	0.07
11	0.79579	0.79579	0.79603 ± 0.0011	0.09
12	0.79472	0.79472	0.79495 ± 0.0010	0.10
31	0.78805	0.78805	0.78841 ± 0.0009	0.71
66	0.78597	0.78597	0.78639 ± 0.0005	3.95
96	0.78541	0.78541	0.78596 ± 0.0005	10.74

On the 16 sausage-type networks, Smith et al. (1995) and Toppur and Smith (2005) decompose the instance into subproblems and then compute a suboptimal tree on each of these sets. They report a value of ρ (which we call "composite ρ") that is actually the average of the ρ values over all the subproblems. Their reporting is not directly commensurable with ours as we report ρ for the entire tree which our heuristic obtains. Therefore, we present the results here not to make direct comparisons, but to demonstrate our approach's ability to handle highly structured instances. While the optimal solution for the sausage-network instances is not known, relative to the conjectured optimal Steiner ratio is 0.78149 (achieved by the infinite-length $\Re$-sausage), Table 4.5 attests that our approach performs well as the average solution obtained is never more than seven percent above this lower bound.

As a final remark, we note that our algorithm typically generates improving candidate trees which have a Steiner topology (and therefore do not require recovery via Algorithm 4.4). For the implemented value of $\chi = 0$, we report in Table 4.6 the percentage of improving candidate trees that require Algorithm 4.4 to recover a Steiner topology (over twenty runs per instance). In addition, we also compute the average percent improvement resulting from recoveries by Algorithm 4.4, where percent improvement for a modification of tree T to tree T' is defined by $(\ell(T) - \ell(T'))/\ell(T)$.

Table 4.6 shows that for the OR-Library instances (Beasley, 1990), fewer than one percent of improving candidate trees do not possess a Steiner topology and Algorithm 4.4 improves these trees by less than 0.5 percent. For the higher-dimension instances of Fampa and Anstreicher (2008), Algorithm 4.4 is required for 4.77, 8.48, and 3.07 percent of the improving candidate trees in $\Re^3, \Re^4$, and $\Re^5$, respectively, with corresponding average percent improvements of 1.28, 1.13, and 1.04 percent. For the two sets of $\Re^3$ instances of Smith et al. (1995) and Toppur and Smith (2005),

Table 4.5: Performance comparison on $\Re^3$ instances appearing in Smith et al. (1995) and Toppur and Smith (2005).

		Toppur and Smith (2005)	Smith et al. (1995)	Randomized Delaunay Triangulation		
Instance	n	Composite ρ	Composite ρ	ρ_{best}	$\bar{\rho} \pm$ st. dev.	CPU (min)
SS2	15	0.8009	0.8345	0.8030	0.8053 ± 0.003	0.2
SS3	23	0.8111	0.8971	0.8069	0.8100 ± 0.002	0.4
SS4	27	0.8130	0.8766	0.8140	0.8203 ± 0.004	0.5
SSM	27	0.8160	0.8950	0.8157	0.8190 ± 0.002	0.6
SA2	21	0.8042	0.8301	0.7966	0.7973 ± 0.001	0.3
SA3	29	0.8218	0.8554	0.8019	0.8053 ± 0.003	0.6
SA4	37	0.8212	0.8611	0.8040	0.8065 ± 0.002	0.9
SAM	54	0.8042	0.8899	0.8309	0.8355 ± 0.002	1.7
LS2	39	0.8029	0.8397	0.7944	0.7964 ± 0.002	1.0
LS3	39	0.8319	0.8478	0.7990	0.8008 ± 0.001	0.9
LS4	43	0.8222	0.8686	0.8048	0.8077 ± 0.002	1.1
LSM	60	0.8229	0.9016	0.8271	0.8302 ± 0.002	2.2
LA2	26	0.8079	0.8465	0.7966	0.7981 ± 0.002	0.4
LA3	37	0.8213	0.8560	0.8026	0.8052 ± 0.002	0.9
LA4	44	0.8181	0.8598	0.8011	0.8030 ± 0.001	1.5
LAM	62	0.8154	0.8763	0.8252	0.8282 ± 0.002	2.2

Table 4.6: We report the percentage of improving candidate trees which require a Steiner topology recovery and the resulting percent improvement in the tree length.

Problem Class		% Recoveries	% Improvement
Beasley (1990)	n		
	10	0.00	0
	20	0.00	0
	30	0.00	0
	40	0.00	0
	50	0.31	0.48
	60	0.74	0.36
	70	0.35	0.31
	80	0.30	0.25
	90	0.16	0.21
	100	0.20	0.26
Fampa and Anstreicher (2008)	d		
	$\Re^3$	4.77	1.28
	$\Re^4$	8.48	1.13
	$\Re^5$	3.07	1.04
Toppur and Smith (2005)		26.23	0.29
Smith et al. (1995)		21.03	0.54

Algorithm 4.4 is required more often (over 20 percent of the improving candidate trees), but the average percent improvement is only 0.54 percent on the sausage instances and 0.29 percent on the network instances. These results indicate our approach is more likely to need Algorithm 4.4 to attain a Steiner topology for a candidate tree in a non-planar instance, but there does not appear to be much improvement to candidate trees with non-Steiner topologies in any dimension.

4.4 Conclusion

We present a heuristic algorithm for the ESTP in $\Re^d$ which utilizes probabilistic insertion of Steiner points in Delaunay simplices. Using a small example, we provide insight on how probabilistic insertion can obtain FSTs on subsets of terminal points that deterministic Steiner insertion schemes may be unable to find. Due to its generality and lack of reliance on dimension-dependent criteria, our approach is agnostic with respect to problem dimension and effectively extends into higher dimensions. Relying on simple mechanisms and second-order cone programming, our algorithm produces competitive solutions within reasonable computational times on both randomized and highly structured instances.

CHAPTER 5
GEOMETRIC CONDITIONS AND SMITH'S ALGORITHM

In this final section, we develop geometric conditions for Steiner minimal trees and describe their implementation into the exact algorithm of Smith (1992). We develop conditions for trees with a Steiner topology, and then more restrictive conditions for trees with a FST. We present computational results on benchmark test problems in $\Re^d$ for $2 \leq d \leq 5$ in Section 5.4.

5.1 Geometric conditions for Steiner minimal trees

Let X be a set of terminal points $\{x_0, ..., x_{n-1}\}$, and assume without loss of generality that x_0 is the origin in $\Re^d$. Denote a set S of Steiner points by $\{s_1, ..., s_k\}$. The *Voronoi diagram* of X is obtained by partitioning Euclidean space into polyhedra with one node x_i in each polyhedra. For every point $x \in X$ the Voronoi region about x, denoted vor(x) consists of all points closer to x than any other point in X; that is vor$(x) = \{u \in \Re^d : \ ||u - x|| \leq ||u - y|| \ \forall \ y \in X\}$. Since the Voronoi regions are polyhedra they have extreme points, which are called *Voronoi points*. Let $V = \{v_1, .., v_m\}$ be the set of Voronoi points. The Voronoi diagram is a natural candidate for developing geometric criteria for SMTs since it exists for $X \subset \Re^d$ for any d, and is itself defined by a condition involving minimal Euclidean norms. The straight line dual of the Voronoi diagram is called the *Delaunay triangulation*. See Section 2.2 for more information on Voronoi diagrams and Delaunay triangulations, which are also shown in Figure 2.1.

Let $B_\delta(x)$ denote the closed ball of radius δ centered at x. The *lune* between two points u and v, denoted $l(u, v)$ is defined as $B_{\|u-v\|}(u) \cap B_{\|u-v\|}(v)$. Lunes are the basis for the following well-known geometric criterion for SMTs, the *lune condition*.

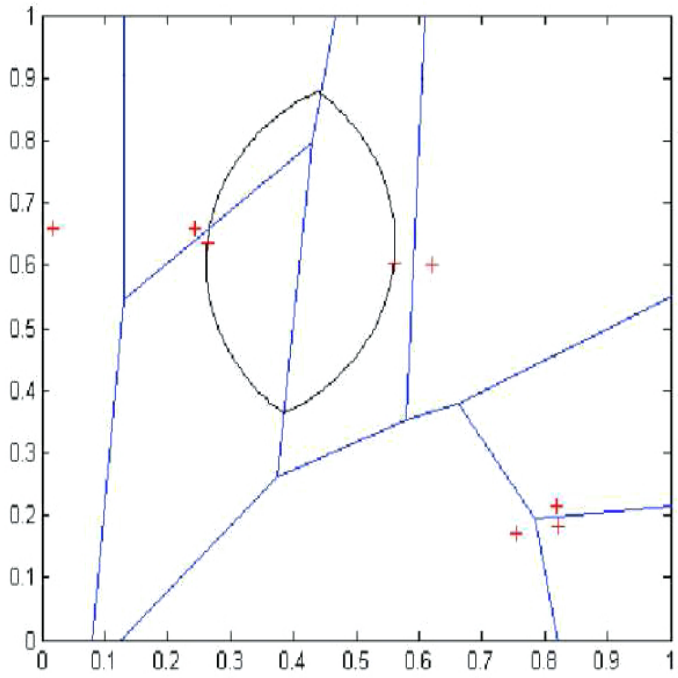


Figure 5.1: A typical lune

Proposition 5.1. *[The Lune Condition] Gilbert and Pollak (1968) If points u and v are connected in the SMT then the interior of $l(u, v)$ contains no other points.*

Proof. If $l(u, v)$ contained another point w we could break the connection between u and v and instead form the connection between w and either u or v, whichever one of these does not disconnect the tree. Either connection will shorten the SMT if w is in the interior of $l(u, v)$. □

The lune condition is an example of a geometric condition that can be used to exclude connections between points in the SMT. The condition is also related to the Delaunay triangulation, as shown in the next lemma.

Lemma 5.2. *If two terminals x_i and x_j are not adjacent in the Delaunay triangulation, then the lune between them contains another terminal point.*

Proof. Let p denote the midpoint of the line segment between x_i and x_j. First note that p is neither in the interior of $\text{vor}(x_i)$ or $\text{vor}(x_j)$. To see this, assume that p

was in the interior of $\text{vor}(x_i)$. Then there would be some q also in the interior of $\text{vor}(x_i)$ such that q was on the line segment between q and x_j. But recall that p was the midpoint of the segment between x_i and x_j, so this means q is closer to x_j than it is to x_i and so could not be in $\text{vor}(x_i)$, a contradiction. Similarly p is not in the interior of $\text{vor}(x_j)$. Therefore there is another terminal z so that $p \in \text{vor}(z)$, $z \neq x_i$, $z \neq x_j$. (If there were no such z then p would be on a boundary face of both $\text{vor}(x_i)$ and $\text{vor}(x_j)$, which is impossible since x_i and x_j are nonadjacent.) We can now show $z \in l(x_i, x_j)$. For example,

$$\|x_i - z\| \leq \|x_i - p\| + \|z - p\| \leq 2\|x_i - p\| = \|x_i - x_j\|,$$

where the first inequality follows from the triangle inequality and the second follows from $p \in \text{vor}(z)$. Likewise, we can show $\|x_j - z\| \leq \|x_i - x_j\|$, and these two inequalities combined give $z \in l(x_i, x_j)$. □

Note: If points X are in general position then midpoint p cannot be on boundary of either $\text{vor}(x_i)$ *or* $\text{vor}(x_j)$*, which makes the second inequality strict, and therefore z is in the interior of the lune.*

Lemma 5.2 immediately implies the following well-known property of SMTs.

Corollary 5.3. *If an optimal Steiner tree contains an edge between terminal points x_i and x_j, then x_i and x_j are adjacent in the Delaunay triangulation.*

5.1.1 Clover regions

Corollary 5.3, which concerns connections between terminal points in an SMT, can also be used to prove a property regarding connections between terminal points and Steiner points.

Lemma 5.4. *Assume that a Steiner point s is connected to a terminal point x in a SMT. Let* $\text{vor}(x)$ *denote the Voronoi region of x in the Voronoi diagram of X. Form the Voronoi diagram of $s \cup X$ treating s as an additional terminal point, and denote the resulting Voronoi region of s by* $\widehat{\text{vor}(s)}$*. Then* $\widehat{\text{vor}(s)} \cap \text{vor}(x) \neq \emptyset$.

Proof. The proof follows from Corollary 5.3 and the fact that a SMT is a minimal spanning tree of $S \cup X$ where the locations of the Steiner points are fixed. □

Note that the condition $\widehat{\text{vor}(s)} \cap \text{vor}(x) \neq \emptyset$ in Lemma 5.4 is equivalent to the hyperplane $\{u : ||u - s|| = ||u - x||\}$ cutting into $\text{vor}(x)$, and since $\text{vor}(x)$ is a polyhedra this hyperplane intersects $\text{vor}(x)$ if and only if $||v - s|| \leq ||v - x||$ for one of the Voronoi points v of $\text{vor}(x)$. This leads us to the concept of a *clover region* associated with x, denoted $\text{clover}(x)$. For convenience take $x = x_0 = 0$, and let the Voronoi points in $\text{vor}(0)$ be $\{v_1, \ldots, v_k\}$. We consider connecting x_0 to a Steiner point s and ask where s may lie so that Lemma 5.4 is satisfied. The perpendicular bisector between 0 and s can be written $\{u \mid s^T u = s^T \left(\frac{s}{2}\right) = \frac{||s||^2}{2}\}$. This perpendicular bisector will "cut off" a particular v_i if $s^T v_i > \frac{||s||^2}{2}$. Then define

$$\begin{aligned} C_i &= \{s \mid ||s||^2 \leq 2s^T v_i\} \\ &= \{s \mid (s^T s - 2s^T v_i + v_i^T v_i) - v_i^T v_i \leq 0\} \\ &= \{s \mid ||s - v_i||^2 \leq ||v_i||^2\} \\ &= B_{\|v_i\|}(v_i). \end{aligned}$$

Thus for each Voronoi point v_i of $\text{vor}(x_0)$, there is an associated ball that describes where a Steiner point can lie to cut off that Voronoi point. We define $\text{clover}(x_0)$ to be the union of these balls over the Voronoi points $\{v_1, \ldots, v_k\}$. We similarly construct $\text{clover}(x_j)$ for each terminal point x_j, $j = 1, \ldots, n-1$. Note that by construction each sphere C_i has the property that it intersects the terminal points x_j such that $v_i \in \text{vor}(x_j)$; in other words, each C_i is the circumsphere of a Delaunay simplex. Thus for each x_j, $\text{clover}(x_j)$ is the union of the circumspheres of the Delaunay simplices that have x_j as a vertex.

The clover region represents a first attempt at using geometry to eliminate topologies that involve Steiner points. In particular, note that if x_i and x_j are connected to the same Steiner point, then their clover regions must intersect. See

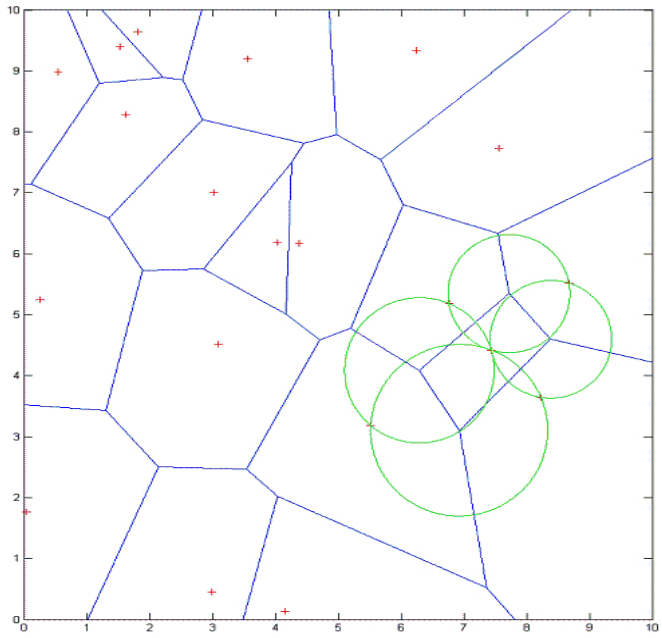


Figure 5.2: A clover region

Figure 5.2 for a picture of a clover region.

5.1.2 Lunar regions

Lemma 5.4, which leads to the definition of a clover region, itself can be viewed as a consequence of the lune condition. A natural question is then if the lune condition can be directly used to give a more restrictive condition for connections between a terminal point and a Steiner point in a SMT. To answer this question, we consider a terminal $x_0 = 0$ connected to a Steiner point s, and determine where the lune property allows s to be. The Steiner point s is allowed to be anywhere such that $l(0, s)$ does not contain any other terminal point x. Considering a particular terminal point x, in order for $x \notin l(0, s)$ we need

- $||x - s|| > \|s\| \Leftrightarrow s^T s < x^T x - 2s^T x + s^T s \Leftrightarrow s^T x \leq \frac{||x||^2}{2}$, or
- $\|x\| > \|s\|$.

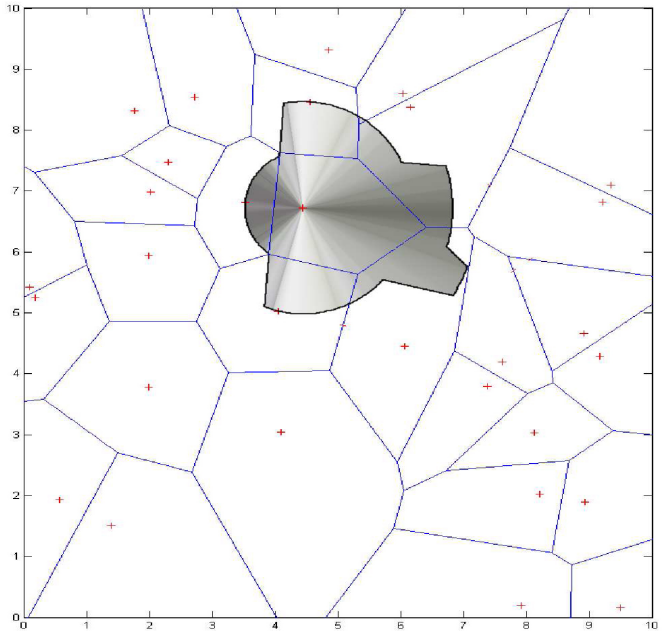


Figure 5.3: A lunar region

The feasible set corresponding to these two regions is the union of a halfspace and a halfsphere, where the hyperplane bounding the halfspace bisects the sphere. There is such a region for each terminal point $x_i \in X/\{0\}$, and taking the intersection of all such regions we arrive at the *lunar region*–a sharper estimate of possible locations for a Steiner point to be connected to a given terminal. Denote the lunar region about a terminal point x as lunar(x). See Figure 5.3 for a an example of a lunar region.

The lunar region–though significantly sharper than the clover region– could be made sharper by considering *extended* Delaunay neighbors as well. Figure 5.4 shows lunar(p) is made sharper by intersecting with the half plane constraint imposed by q, even though p and q are distance two apart in the Delaunay triangulation.

As with clover regions, lunar regions can be used to exclude possible Steiner topologies; if lunar(x_i) does not overlap lunar(x_j), then one need not consider any topology in which x_i and x_j connect to the same Steiner point. Because lunar regions

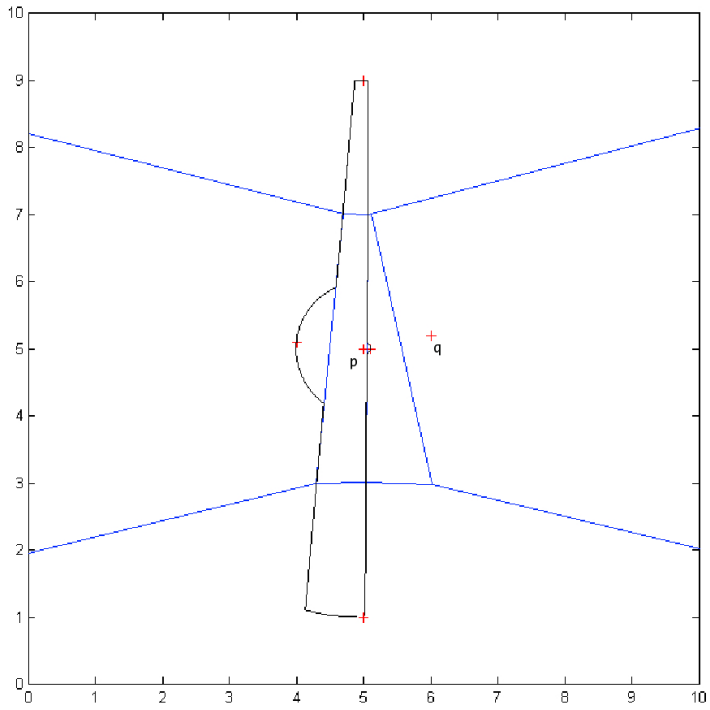


Figure 5.4: Extended Delaunay neighbor q can sharpen lunar(p).

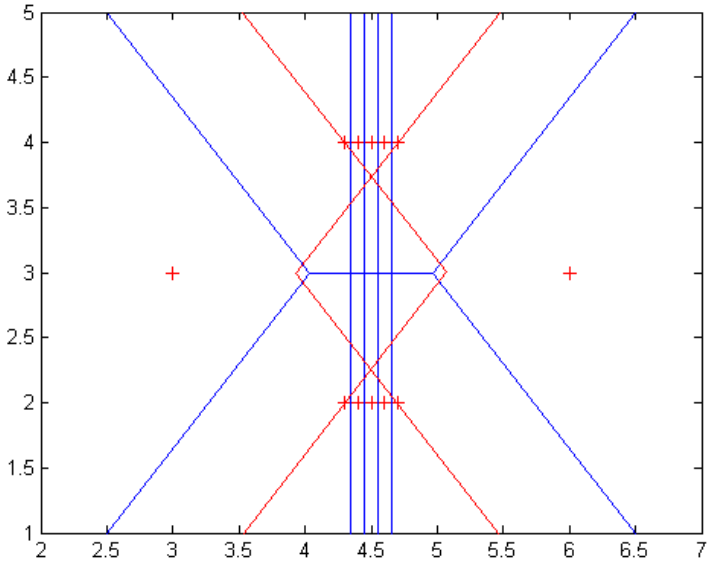


Figure 5.5: Delaunay distances–no matter how large–do not imply disjoint lunar regions.

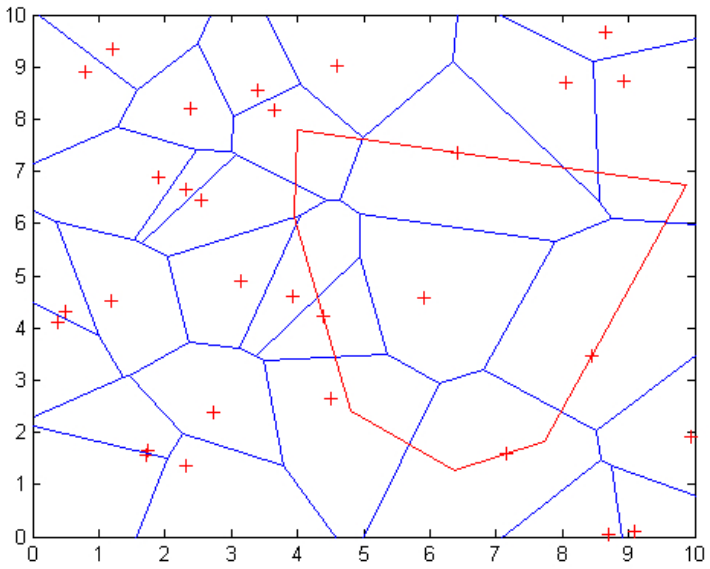


Figure 5.6: A doubled Voronoi cell

are nonconvex, determining whether or not two such regions overlap requires case checking. We next show that there is an easily-computed convex relaxation of the lunar regions.

5.1.3 Doubled Voronoi cells

The doubled Voronoi cell about x, denoted $\mathrm{dvor}(x)$, is simply $\mathrm{vor}(x)$ dilated by a factor of two (see Figure 5.6). Note that each semi-spherical portion of the lunar region is tangent to the corresponding bounding hyperplane of the doubled Voronoi cell, as shown in Figure 5.7. As an immediate consequence of this fact we have the following relationship between lunar regions and doubled Voronoi cells.

Lemma 5.5. *For any terminal point x,* $\mathrm{vor}(x) \subset \mathrm{lunar}(x) \subset \mathrm{dvor}(x)$.

Proof. Assume without loss of generality that $x = x_0 = 0$. Let x_i, $i = 1, \ldots, k$ be the neighboring terminal points in the Delaunay triangulation. Then

$$\mathrm{dvor}(x_0) = \{x \,|\, x_i^T x \leq \|x_i\|^2,\ i = 1, \ldots, k\}.$$

The lunar region, $\mathrm{lunar}(x_0)$, is the set of x such that for $i = 1, \ldots, k$ either $x_i^T x \leq \|x_i\|^2/2$ or $\|x\| \leq \|x_i\|$. Note that for each i, the first constraint defines the face

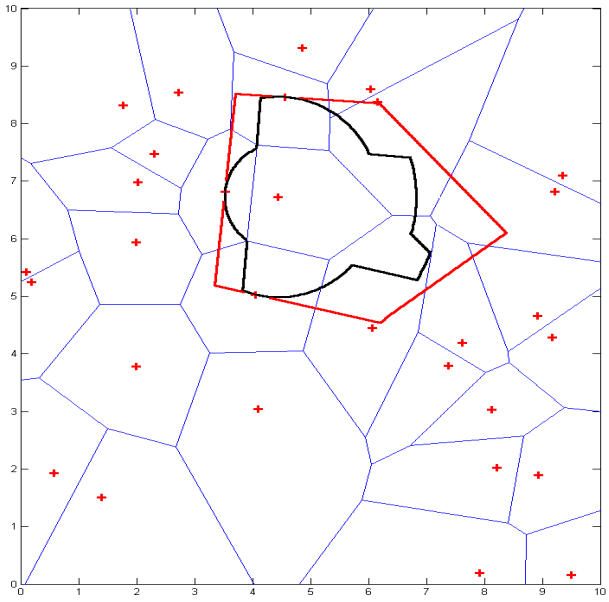


Figure 5.7: The lunar region is contained in the doubled Voronoi cell.

of vor(0) corresponding to x_i, and therefore $\text{vor}(x_0) \subset \text{lunar}(x_0)$. Moreover if x satisfies the second condition for a given i, then

$$x_i^T x \leq \|x_i\|\|x\| \leq \|x\|^2.$$

It follows immediately that $x \in \text{dvor}(x_0)$ for any $x \in \text{lunar}(x_0)$. □

See Figure 5.8 for motivation of the following theorem, whose proof is due to De la Mora (2007). Theorem 5.6 and Lemma 5.5 together imply the appealing hierarchy

$$\text{vor}(x) \subset \text{lunar}(x) \subset \text{dvor}(x) \subset \text{clover}(x)$$

for any terminal point x.

Theorem 5.6. *For any terminal point x,* $\text{dvor}(x) \subset \text{clover}(x)$

Proof. For simplicity we take $x = x_0 = 0$. Each Delaunay simplex that has x_0 as a vertex has an additional d vertices, each of which is a terminal point. The cones generated by these simplices partition $\Re^d$, so any $u \in \text{vor}(0)$ must be in one such cone. Assume that u is in the cone generated by x_i, $i = 1, \ldots, d$, so there are $\lambda_i \geq 0$,

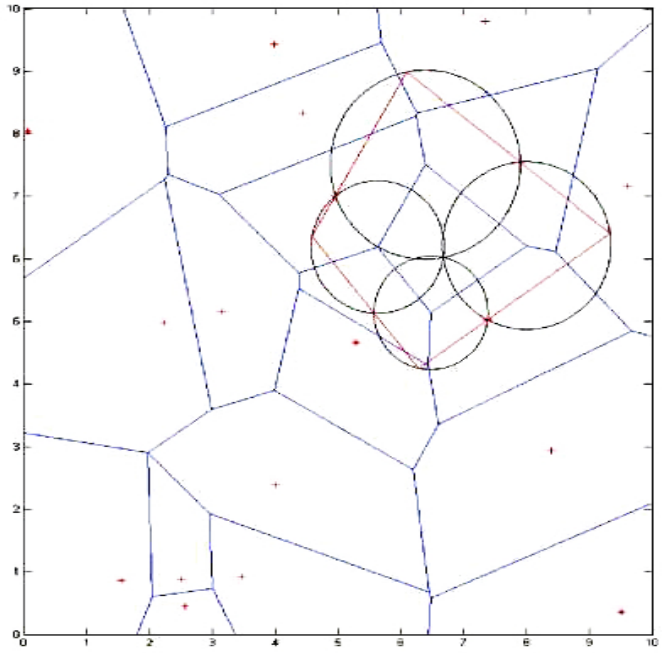


Figure 5.8: The doubled Voronoi region is contained in the clover region.

$i = 1, \ldots, d$ such that

$$\sum_{i=1}^{d} \lambda_i x_i = u.$$

Let v be the Voronoi point that is the center of the hypersphere that circumscribes the simplex with extreme points x_i, $i = 0, 1, \ldots, d$. It follows that $\|x_i - v\| = \|v\|$, $i = 1, \ldots, d$, which is equivalent to

$$\|x_i\|^2 = 2x_i^T v, \quad i = 1, \ldots, d. \tag{5.1}$$

Our goal is to show that $2u \in \text{clover}(0)$. To do this it suffices to show that

$$||2u - v|| \leq ||v||,$$

which is equivalent to $u^T u \leq u^T v$. Since $u \in \text{vor}(0)$ we have $x_i^T u \leq \|x_i\|^2/2$, $i = 1, \ldots, d$, which combined with (5.1) implies that $x_i^T u \leq x_i^T v$, $i = 1, \ldots, d$. Therefore

$$u^T u = \sum_i \lambda_i x_i^T u \leq \sum_i \lambda_i x_i^T v = u^T v,$$

as required. □

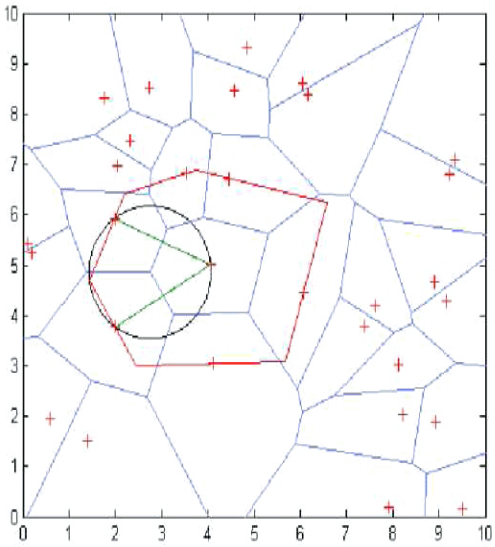


Figure 5.9: Proving the doubled Voronoi region is contained in the clover region

A small counterexample demonstrates that a right handed inclusion in Theorem 5.6 is impossible. That is, there does not exist an $\alpha \in \Re$ such that clover$(p) \subset \alpha \cdot \text{vor}(p)$. The construction of a counterexample is shown in the sequence of figures in Table 5.1.3. As point q moves toward point p, the Voronoi face separating p and q moves arbitrarily close to p while the circumsphere at v approaches some fixed radius. Thus, for any $\alpha > 0$ it is possible to move q close enough to p to ensure the circumsphere at v is not contained in the region defined by $\alpha \cdot \text{vor}(p)$. The quadruple Voronoi region has been shown in the right hand figures for illustrative purposes.

5.2 Geometric conditions for SMTs with a FST

In this section, we provide more restrictive conditions for SMTs with a FST. In Section 5.3, we incorporate these conditions in the scheme of Smith (1992).

5.2.1 Smallest spheres

Clover regions, doubled Voronoi cells, and lunar regions provide an increasingly sharp restriction on the location of a Steiner point connected to a given terminal. In Smith's scheme all generated trees have a FST, and so all terminal points

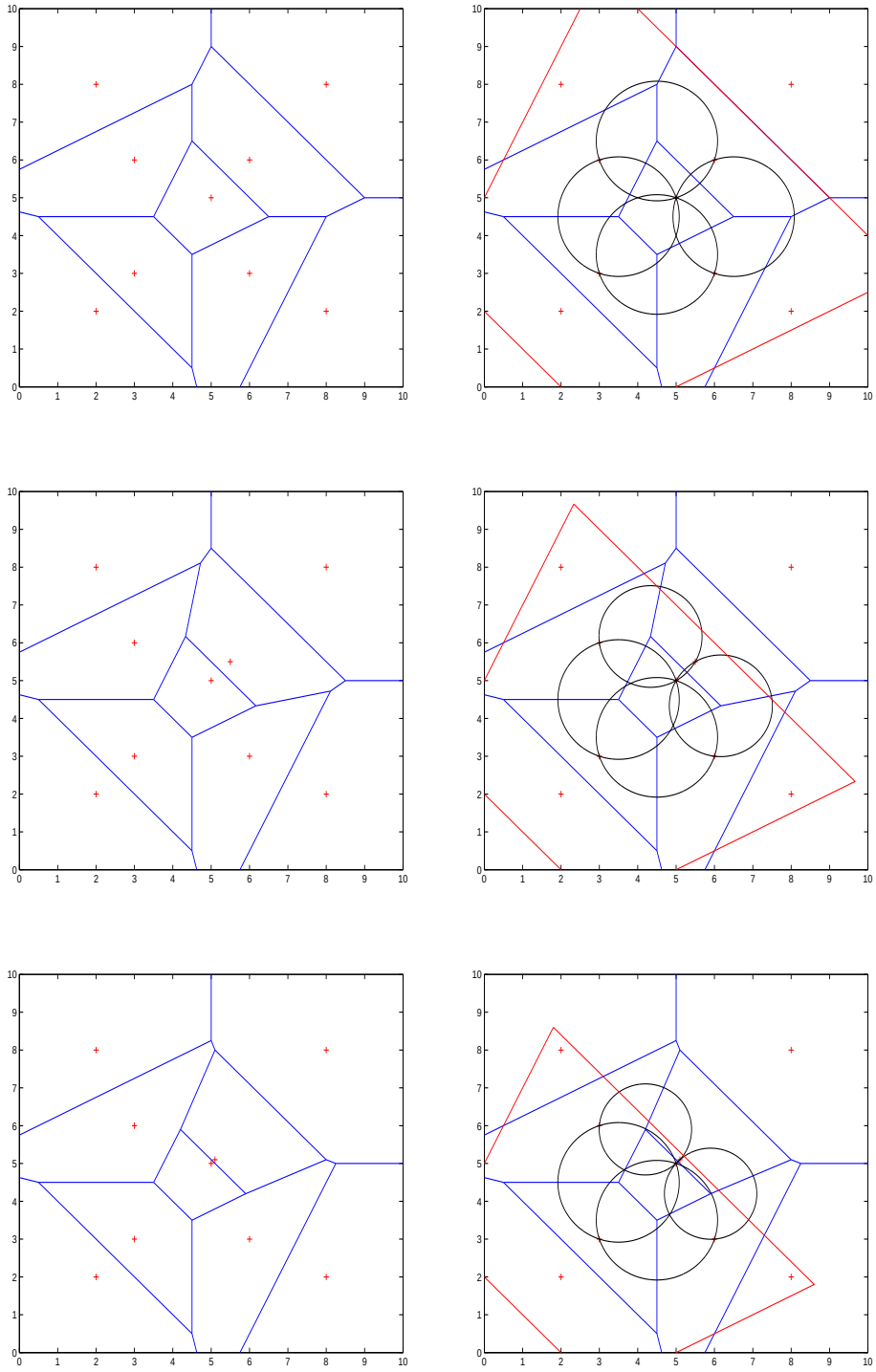


Table 5.1: The clover region is not contained in any multiple of the Voronoi cell: as the two points move closer together, the circumspheres approach a fixed radius while $\alpha \cdot \mathrm{vor}(p)$ is made arbitrarily close to $\mathrm{vor}(p)$.

are leaves of the tree. This allows the feasible locations of Steiner points to be further restricted to the "smallest sphere" (which is contained in the lunar region). A simple edge exchange argument proves the result:

Lemma 5.7. *In an SMT with a FST, an edge between a Steiner point and terminal point, x_i, in the SMT is of length at most d_i, where d_i is the distance from x_i to the nearest other terminal point.*

Proof. If an edge longer than d_i where connected to terminal x_i, remove it and break the SMT into two connected components: x_i and the rest of the tree. Connect x_i to the nearest terminal with an edge of length d_i and shorten the tree. □

Call $B_{d_i}(x_i)$ the "smallest sphere" about x_i. We conclude from Lemma 5.7 that two terminals x_i and x_j may be connected to a common Steiner point if

$$||x_i - x_j|| \leq d_i + d_j.$$

We strengthen this condition by incorporating the angle condition.

Lemma 5.8. *Consider terminals x_i and x_j with smallest spheres of radii d_i and d_j, respectively. Then x_i and x_j may be connected to a common Steiner point, s, if*

$$||x_i - x_j|| \leq \sqrt{d_i^2 + d_j^2 + d_i d_j}.$$

Proof. Consider the angle, $\angle x_i s x_j$. By the law of cosines:

$$||x_i - x_j||^2 = ||x_i - s||^2 + ||s - x_j||^2 - 2||x_i - s|| \cdot ||s - x_j|| \cos\theta.$$

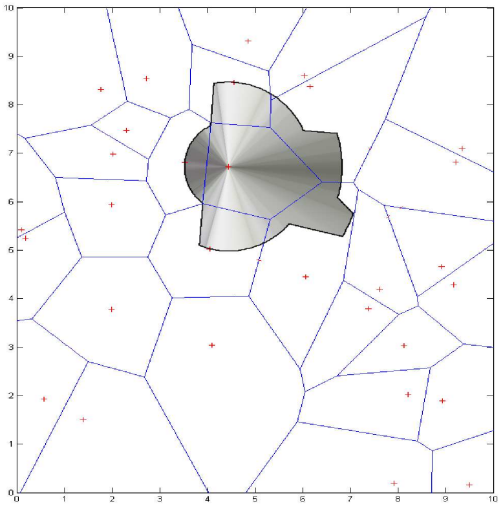
Since we have a SMT with a FST, $\theta = 120°$:

$$||x_i - x_j||^2 = ||x_i - s||^2 + ||s - x_j||^2 + ||x_i - s|| \cdot ||s - x_j||.$$

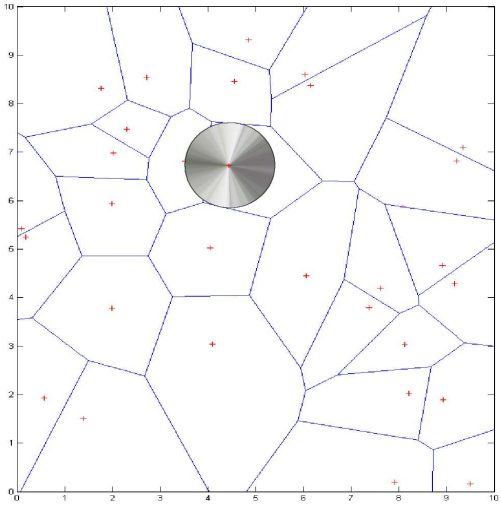
Apply Lemma 5.7 which gives $||x_i - s|| \leq d_i$ and $||x_j - s|| \leq d_j$.

□

We extend this result to more than one Steiner point; recall the upper bound on lengths of edges developed in Section 2.1. Given two terminals x_i and x_j, let b_{ij} denote the length of the longest edge on the unique path between x_i and x_j in the MST (the *bottleneck distance*. The bottleneck distance is an upper bound on the



(a) The *lunar region* contains the possible locations for a Steiner point connected to the given terminal point to not violate the lune property.



(b) For Smith's algorithm, the feasible region for the location of a Steiner point attached to a terminal point shrinks to the "smallest sphere".

Figure 5.10: The *lunar region* 5.10(a) and the "smallest sphere" 5.10(b) restrict the locations for Steiner points attached to a terminal point for an arbitrary SMT and an SMT with a FST, respectively.

length of edges on the path between x_i and x_j in the SMT (see Lemma 2.2. Note that $b \geq d_i$ for all terminals, x_i. We conclude from Lemma 2.2 that x_i and x_j may be connected by two or fewer Steiner points if

$$||x_i - x_j|| \leq d_i + d_j + b_{ij}.$$

Once again, we strengthen this condition by applying the angle condition.

Lemma 5.9. *Two terminals x_i and x_j may be connected by two or fewer Steiner points if*

$$||x_i - x_j|| \leq \sqrt{(d_i + d_j)^2 + b_{ij}^2 + (d_i + d_j)b_{ij}}.$$

Proof. Consider a configuration on points (x_i, s_1, s_2, x_j), where the edges obey the angle condition and length restrictions in Lemma 5.7. Note that increasing the length of any edge only increases the inter-terminal distance. Without loss of generality, assume all edge lengths are at their upper bounds.

Let $(x_i, s_1, s_2, \tilde{x_j})$ denote the configuration where the last point has been rotated into the plane defined by the first three points, maintaining $\angle s_1 s_2 \tilde{x_j}$. There are two choices for point $\tilde{x_j}$ which satisfy the angle condition. Choose $\tilde{x_j}$ on the opposite side of edge (s_1, s_2) from x_i, see Figure 5.11. Since $b \geq d_i$ and $b \geq d_j$, the line segment between $(x_i, \tilde{x_j})$ must intersect edge (s_1, s_2). Let m be this point of intersection. Then:

$$\begin{aligned} ||x_i - \tilde{x_j}|| &= ||x_i - m|| + ||m - \tilde{x_j}|| \\ &= ||x_i - m|| + ||m - x_j|| \\ &\geq ||x_i - x_j|| \end{aligned}$$

where the second equality follows since $\tilde{x_j}$ and x_j are the same distance from any point on the line between (s_1, s_2). We have shown the planar configuration does not decrease the inter-terminal distance.

The inter-terminal distance of the planar configuration is easily found since alternating edges are parallel. Translate and apply the law of cosines with $\theta = 120°$:

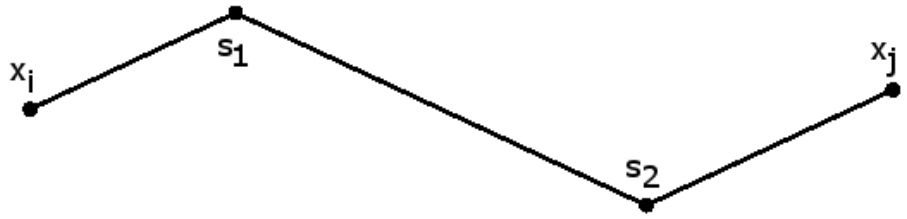


Figure 5.11: The configuration with two Steiner points which maximizes the inter-terminal distance lies in a plane and has alternating edges.

$$||x_i - x_j|| \leq \sqrt{(d_i + d_j)^2 + b_{ij}^2 + (d_i + d_j)b_{ij}}.$$

□

We have shown that if two terminals are "too far" apart, they must be connected by one or more Steiner points. We conjecture an extension of this result to k points ($k-2$ Steiner points). The conjecture follows if the configuration which maximizes the inter-terminal distance can be proved to lie in a plane.

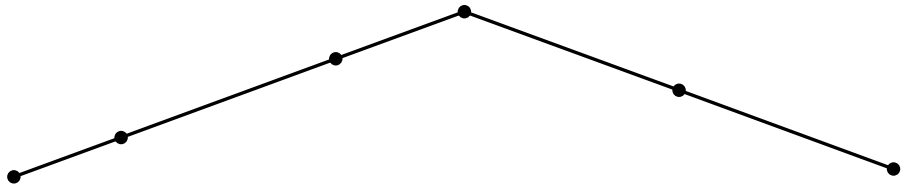
Conjecture 5.10. *The minimum number of Steiner points on the path between terminals x_i and x_j in the SMT is given by the minimum integer $k \geq 1$ such that*

$$d(x_i, x_j) \leq \begin{cases} \sqrt{(d_i + \frac{k-1}{2}b_{ij})^2 + (d_j + \frac{k-1}{2}b_{ij})^2 + (d_i + \frac{k-1}{2}b_{ij})(d_j + \frac{k-1}{2}b_{ij})} & \text{for } k \text{ odd,} \\ \sqrt{(d_i + d_j + \frac{k-2}{2}b_{ij})^2 + (\frac{k}{2}b_{ij})^2 + (d_i + d_j + \frac{k-2}{2}b_{ij})(\frac{k}{2}b_{ij})} & \text{for } k \text{ even.} \end{cases}$$

5.3 Implementation of geometric conditions

We have presented geometric criteria for SMTs and SMTs with an FST, and we now implement the criteria into Smith's scheme. One of the difficulties with Smith's scheme is that parent topologies with a "bad" configuration can have "good" descendants due to the constructive nature of the merge operation (see Section 3.3.2). We quantify the ability of the merge operation to correct violated criteria: each

(a) The configuration which is conjectured to maximize the inter-terminal distance



(b) Alternate edges are parallel, which allows easy computation of the inter-terminal distance

Figure 5.12: The conjecture for many Steiner points

merge operation increases the number of Steiner points on the path between terminals by one. If two terminals are connected by fewer than the required number of Steiner points, (e.g. they are connected by one Steiner point, but three Steiner points are required by Lemma 5.9), then more merges are needed to correct this violated geometric condition (e.g. 3-1=2 more merges are needed). If there are not enough merges remaining in the branch-and-bound scheme to correct all violations, this node may be removed from consideration.

Lemmas 5.7 and 5.9 are used to create a matrix $D_{n,n}$ in a pre-processing step whose entries are limited to the set $\{1, 2, 3\}$, indicating the *minimum* number of Steiner points on the path from terminals x_i to x_j in the SMT. We tabulate the *deficit* for a partial FST encountered at level k of Smith's scheme as follows. If two terminals x_i and x_j are connected by m Steiner points and $m < D(i, j)$, then we add $D(i, j) - m$ to the deficit count for the topology. We sum the deficits over all disjoint paths in the partial FST. If the number of deficits is greater than the number of merges remaining, this topology and all its descendants may be removed from consideration (with no need to compute its length).

Note that the deficit tabulation is additive over disjoint paths. If P_1 and P_2 are disjoint paths between two pairs of terminals, each one requiring m_1 and m_2 more merges, respectively, then $m_1 + m_2$ merges are required in total to correct this violation. This follows since each merge can add one Steiner point to either (but not both) of the paths. A sub-routine for packing of disjoint paths is needed since one merge can correct multiple violations on paths which share an edge.

We perform a greedy set-packing of disjoint paths. Sort all paths by their deficit value, $D(i,j) - m$, from highest to lowest (currently, the values are 1 and 2). We would like to pack as many short paths as possible, so first choose all paths of length two (that is, paths with two edges) with deficit 2. Then choose all paths of length two with deficit 1. Finally, pack the paths of length three which have deficit 1. Note there are no paths of length three with deficit two, since the entries of the matrix D range from 1 to 3. Calculate the deficit count for the FST by summing deficits over the disjoint paths.

We incorporate this new fathoming code immediately following the initial bound check on Line 4 in Algorithm 2.4. We do so after checking the parent bound, since computing deficit information requires more computation than checking the bound. If the deficit count is greater than than the merges remaining in Smith's scheme, this FST may be fathomed with no need to compute its length.

We illustrate the entries of the matrix D across randomized instances of size $N = 10, 15, 20, 25$ in $\Re^3$, the results are shown in Figure 5.2.1. When the number of terminals is small ($N = 10$), there are many 1's and fewer 2's or 3's. As N increases, the number of 1's decreases and the number of 2's and 3's increases. This is promising for integrating the results into Smith's scheme, since two terminals will always be connected by at least one Steiner point. Only entries larger than 1 provide additional criteria for fathoming FSTs.

We make another note on Conjecture 5.10. Let us assume the conjecture is

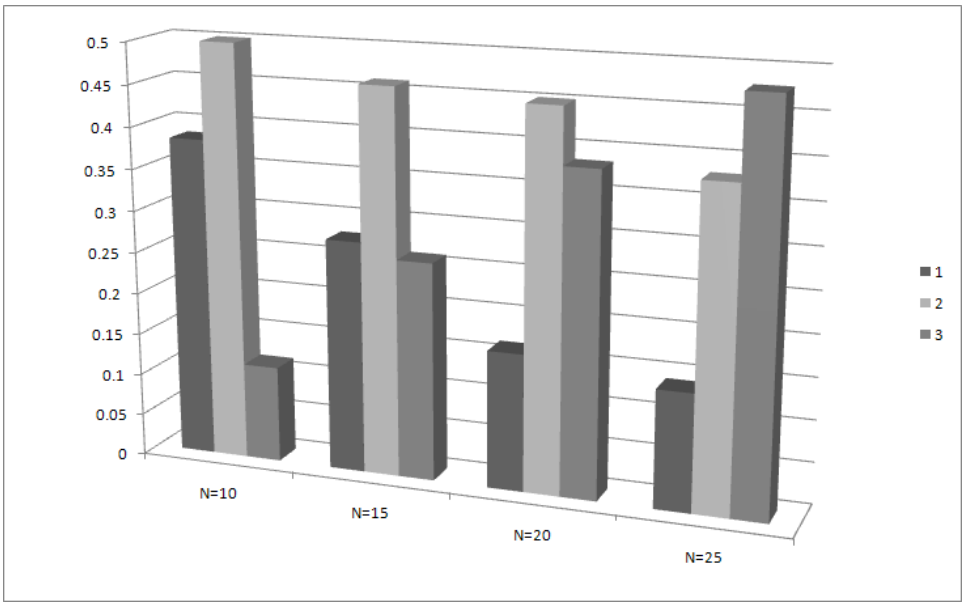


Figure 5.13: The distribution of the minimum number of Steiner points connecting terminal points in $\Re^3$ as the number of terminals, N, increases. The number of 3's increases with N.

true and create the matrix $D(i, j)$. Across all instances appearing in Fampa and Anstreicher (2008) ($N = 10$ in $\Re^d, d = 2, ..., 5$) *no* new entries appeared. No new D matrix entries appeared in the instances of size 12 in $\Re^3$. Three of the ten instances of size 14 in $\Re^3$ contained one or more 4's. Two of these three instances contained only one 4 in the D matrix. Four of the five instances of size 16 in $\Re^3$ contained one or more 4's.

5.4 Computational results

In this section, we present computational results for our geometric enhancements of the scheme of Smith (1992). The nodes in the branch-and-bound (B&B) tree are full Steiner topologies on subsets of terminal points, and children are created by merging a new terminal point to each edge in the existing tree. We refer to *nodes processed* as the number of full Steiner topologies in the B&B tree which are pulled from the list and not immediately fathomed based on length or geometric conditions. That is, they will require a call to Mosek (Andersen and Andersen, 2010)

to optimize the location of Steiner points and determine the length of the FST. We denote by *CPU-time* the CPU seconds taken to solve an instance. All values in tables represent averages taken over the number of instances solved. The instances of Fampa and Anstreicher (2008) are randomized instances in the hypercube $[0, 10]^d$ with $N = 10$ terminals. For $d = 2, 4, 5$ there are 5 instances each, and for $d = 3$ there are 10 instances. We generate randomized instances in $\Re^3$ each with ten instances distributed in the unit hypercube, with $N = 12, 14, 16$. All computational results in this section are variations of the C implementation of Smith's algorithm; the code for the original Smith algorithm is provided by Marcia Fampa. All runs are performed on a dual core Pentium 3.20GHz processor with 2 GB of ram, operating a Linux platform.

5.4.1 The effect of sorting the instance

Fampa and Anstreicher (2008) utilize a "strong branching" scheme to vary the order in which terminals are merged to the existing partial FST. The next terminal merged is chosen in such a way that the maximum number of children will be fathomed at the next level. The advantage of the strong branching scheme is that the created children FSTs are more likely to be fathomed, resulting in reduced computational effort and number of nodes processed. The disadvantage of the strong branching scheme is the computation required to decide the next terminal to merge. We investigate the role of sorting the input instance as a way of accelerating fathoming without the computational effort associated with a sophisticated branching scheme. We sort the terminals via distance from their centroid, with the first terminal being the farthest away and the final terminal being the closest. The hope is that by first merging terminals which are "far apart" we grow the length of the tree as fast as possible. We compare our results on the same instances as presented in Fampa and Anstreicher (2008).

As Table 5.2 demonstrates, sorting the terminal points significantly reduces

the nodes processed and CPU-time required. *Nodes-factor* refers to the factor of reduction in nodes processed in the original Smith algorithm compared with the sorting or the strong-branching; the calculation is analogous for *time-factor*. Sorting the terminals via centroid results in average nodes factors of between 3.5 and 15.9 compared with factors ranging from 26.9 to 160.2 for the strong branching scheme of Fampa and Anstreicher (2008). As expected, the more sophisticated strong-branching scheme reduces the nodes processed by a much larger factor. The CPU-time required by the algorithm is vastly reduced by sorting, with factors comparable to Fampa and Anstreicher (2008). Sorting causes more nodes to be processed, but it avoids the additional computational effort from solving the sub-problems associated with choosing the next terminal to merge. The sorting (done in a pre-processing step) does not require any additional time to solve created sub-problems as in Fampa and Anstreicher (2008). Sorting via centroid presents an economical way to reduce both nodes process and computational time; the results presented throughout the rest of this thesis are sorted via centroid.

5.4.2 The effect of initial upper bounds

The branch-and-bound algorithm must be initialized with some value for the upper bound (UB). For ease, ∞ may be used. For the ESTP, the length of the MST or any other spanning tree may also be used. Smith (1992) comments that "the backtracking algorithm itself soon generates a very good upper bound." We test this by initializing the BB algorithm with an upper bound equal to the best value found by the heuristic algorithm in Chapter 4. We do not include the computation time taken to produce the heuristic solution. We modify line 1 of Algorithm 2.4 by setting UB equal to the best found value of the heuristic. As Table 5.3 demonstrates, the improvement is marginal and comes at the cost of the extra run-time of the heuristic algorithm in Chapter 4.

Table 5.2: We report the effect of sorting the terminal points on randomized instances in $\Re^d$, $2 \leq d \leq 5$ with $n = 10$.

	Smith (1992)		Smith (1992) w/ sorting				Fampa and Anstreicher (2008)	
	Nodes	CPU-time	Nodes	CPU-time	Nodes	Time	Nodes	Time
d					Factor	Factor	Factor	Factor
2	16821.6	25.04	1060.2	1.6	15.9	15.7	160.2	10.5
3	139971.4	248.6	39570.8	69.9	3.5	3.6	50.7	4.8
4	368762.8	766.5	60488.6	120.1	6.1	6.4	26.9	2.8
5	470321.8	1116	128484.6	296.3	3.7	3.8	50.8	4.4

Table 5.3: We report the effect of initial upper bounds on randomized instances in $\Re^d$, $2 \leq d \leq 5$.

Problem Class							
Fampa and		Smith (1992)		Initial upper bound			
Anstreicher (2008)	d	Nodes	CPU-time	Nodes	CPU-time	Nodes Factor	Time Factor
	$\Re^2$	1060.2	1.6	868.8	1.31	1.2	1.2
	$\Re^3$	39570.8	69.9	27524.6	48.27	1.4	1.4
	$\Re^4$	60488.6	120.1	56337.2	112.3	1.1	1.1
	$\Re^5$	128484.6	296.3	118998.8	273.6	1.1	1.1
Randomized in $\Re^3$	n						
	12	323829.2	613.3	309782.2	583.4	1.0	1.1
	14	3053659.2	6239.0	2868429.7	5836	1.1	1.1
	16	21114697.6	48350.0	16289236.6	36983.6	1.3	1.3

5.4.3 The effect of geometry

We report the effect of adding fathoming by geometry to the algorithm of Smith (1992). As Table 5.4 demonstrates, geometric conditions have a significant impact on fathoming of nodes and computation time. All instances in $\Re^d$, $d > 2$, show a significant decrease in nodes processed and time required. We note that these two factors are typically equal, due to the minor computational effort in checking geometric criteria (compared to calling Mosek). The instances in $\Re^5$, for example, averaged 167 CPU-seconds per instance with 165.9 seconds in calls to Mosek.

The performance in $\Re^3$ appears to be monotone decreasing with values of n. It is unclear whether this is the limit of the geometric conditions or the unresolved Conjecture 5.10. We test this by assuming the truth of the Conjecture 5.10 and running the instances of size 14 and 16 and $\Re^3$, the results are shown in Table 5.6. The instances of size 14 demonstrate a small improvement in nodes processed, but the improvement is unobservable when rounding Nodes Factor and Time Factor to two decimal places. The instances of size 16 demonstrate a small improvement in the nodes processed and computation time.

5.4.4 The effect of geometry and initial upper bounds

In this section, we combine the geometric conditions and initial upper bounds to determine the marginal improvement to the exact algorithm. These two combined effects will improve the algorithmic performance, but it is unclear how many nodes fathomed by geometry would also be fathomed by length. FSTs which violate geometric criteria are also likely to be "long," and thus fathomable by length. As Table 5.5 shows, there is only a marginal improvement by also including the initial upper bounds. The performance of the algorithm is no longer monotone decreasing with respect to instance size in $\Re^3$. This could be attributed to good quality heuristic

values for instances of size 16, as a similar trend is observed when utilizing just the upper bound. Due to the significant running time of the heuristic (which we do not include), the geometric criteria provide the most improvement to the algorithm with the least computational effort.

We further illustrate the performance of the algorithm on the instances of size 10 in $\Re^3$ as seen in Table 5.7. Note that instance number 6, the geometric conditions *increased* the number of nodes processed. In this instance, geometric conditions caused a node to fathom which would have otherwise improved the upper bound value. Without improving this upper bound, additional nodes that could not be fathomed by geometry continued to be processed. When combined with the initial upper bounds, we see a reduction in the number of nodes processed. These results indicate that a reasonable bound should be obtained before fathoming by geometry. Note also that the effect of geometry is most noticeable on the most difficult instances 1 and 10.

5.5 Conclusion

We have presented geometric criteria which can be used to eliminate candidate topologies in implicit enumeration algorithms. We first presented criteria for SMTs with an arbitrary topology, then for SMTs with an FST. Empirical evidence indicates that such criteria are useful in an exact algorithm to promote fathoming. The use of initial upper bounds in the algorithm was found to have a marginal effect, confirming Smith's comment that the algorithm quickly produces good solutions. Fathoming was only marginally improved by combining the geometric conditions and initial upper bounds. Merging the terminals via distance from centroid was shown to reduce computation time on the order of a more sophisticated strong branching scheme. Future work includes incorporating branching based on geometric conditions, i.e., merging in such a way to create the most violations to promote fathoming higher in the branch-and-bound tree.

Table 5.4: We report the effect of geometric conditions on randomized instances in $\Re^d$, $2 \leq d \leq 5$.

Problem Class							
Fampa and		Smith (1992)		With geometry			
Anstreicher (2008)	d	Nodes	CPU-time	Nodes	CPU-time	Nodes Factor	Time Factor
	$\Re^2$	1060.2	1.6	728.0	1.6	1.5	1.0
	$\Re^3$	39570.8	69.9	22765.7	40.2	1.7	1.7
	$\Re^4$	60488.6	120.1	44858.2	89.9	1.3	1.3
	$\Re^5$	128484.6	296.3	72849.8	167.0	1.8	1.8
Randomized in $\Re^3$	n						
	12	323829.2	613.3	191318.8	362.7	1.7	1.7
	14	3053659.2	6239.0	2284116.9	4681.0	1.3	1.3
	16	21114697.6	48350.0	15849345.2	36200.0	1.3	1.3

Table 5.5: We report the effect of geometric conditions combined with initial upper bounds on randomized instances in $\Re^d$, $2 \leq d \leq 5$.

Problem Class							
Fampa and		Smith (1992)		Geom & IUB			
Anstreicher (2008)	d	Nodes	CPU-time	Nodes	CPU-time	Nodes Factor	Time Factor
	$\Re^2$	1060.2	1.6	588.2	0.902	1.8	1.8
	$\Re^3$	39570.8	69.9	18090.4	31.59	2.2	2.2
	$\Re^4$	60488.6	120.1	42090.4	83.81	1.4	1.4
	$\Re^5$	128484.6	296.3	66434.4	152.1	1.9	1.9
Randomized in $\Re^3$	n						
	12	323829.2	613.3	185948.1	350.7	1.7	1.7
	14	3053659.2	6239.0	2172100.3	4451	1.4	1.4
	16	21114697.6	48350.0	12875306.4	28670.6	1.6	1.7

Table 5.6: We assume the truth of the Conjecture 5.10 and test instances of size 14 and 16 in $\Re^3$.

* =conjecture assumed					
		With geometry		With geometry*	
		Nodes Factor	Time Factor	Nodes Factor	Time Factor
Randomized in $\Re^3$	n				
	14	1.34	1.33	1.34	1.33
	16	1.33	1.34	1.37	1.37

Table 5.7: We investigate further the instance of size $n = 10$ in $\Re^3$ of Fampa and Anstreicher (2008).

	Smith	IUB		Geometry		IUB & Geom	
	Nodes	Nodes	Factor	Nodes	Factor	Nodes	Factor
Instance							
1	58866	39436	1.49	29557	1.99	23187	2.54
2	4066	4066	1.00	2544	1.60	2535	1.60
3	1870	1825	1.02	1702	1.10	1550	1.21
4	3087	2420	1.28	2786	1.11	2237	1.38
5	20101	18801	1.07	16609	1.21	15620	1.29
6	838	482	1.74	1037	0.81	475	1.76
7	3450	3305	1.04	3372	1.02	3119	1.11
8	52390	45810	1.14	37839	1.38	34417	1.52
9	14707	10928	1.35	13822	1.06	10138	1.45
10	236333	148173	1.59	118389	2.00	87626	2.70

APPENDIX
COUNTING TYPES OF STEINER POINTS

In this section, we classify the types of Steiner points based on the adjacent point (terminal or Steiner) in the tree. We determine the number of type two, one, and zero Steiner points at each level of Smith's enumeration scheme (adjacent to two terminals, one terminal, and zero terminals, respectively). We first count the Steiner points descendant of one partial full Steiner topology at level k. We then sum over all partial FSTs at level k and divide by the number of FSTs at level $k+1$ to obtain an average per tree count of varying type Steiner points at any level in the scheme. For level $k > 1$ of Smith's scheme, there are no type three Steiner points so they are omitted from the discussion.

A.1 The descendants of a parent topology

Consider a partial FST at level $k > 1$ of Smith's enumeration scheme. Given the number of each type of Steiner point in the tree, we count the number of each type of Steiner points among all of its children. Consider a merge operation applied to each of the $2k + 1$ edges of this parent.

Consider first merging to any Steiner-Steiner edge of the parent topology. Every such merge increases the number of type one Steiner points by one, while preserving the numbers of all other types of Steiner points. See Figure A.1. Consider merging to the Steiner-terminal edge adjacent to a type one Steiner point. Such a merge increases the number of type two and type zero Steiner points by one and decreases the number of type one Steiner points by one, see Figure A.2. Consider next merges applied to Steiner-terminal edges adjacent to type two Steiner points, which preserves the number of type zero and type two Steiner points and increases the number of type one Steiner points by one, see Figure A.3.

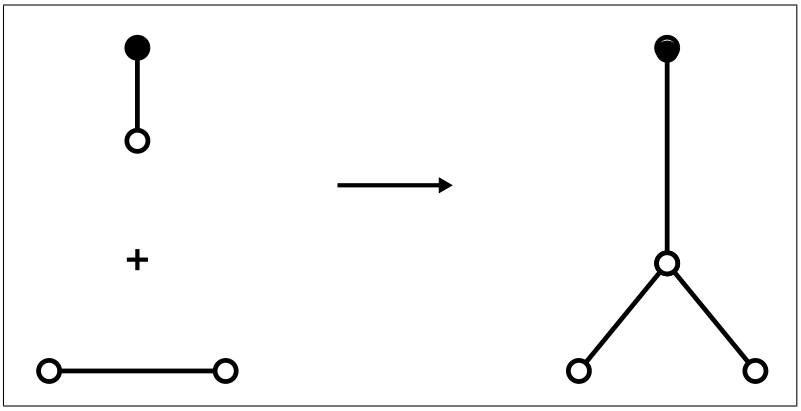


Figure A.1: Merging to a Steiner-Steiner edge always increases the number of type one Steiner points by one. (Steiner points are shown in white, terminal node in black.)

Table A.1: The result of merging to Steiner-Steiner and Steiner-terminal edges.

Steiner-Steiner			Type One S-T			Type Two S-T		
n_2^{k+1}	$\leftarrow$	n_2^k	n_2^{k+1}	$\leftarrow$	$n_2^k + 1$	n_2^{k+1}	$\leftarrow$	n_2^k
n_1^{k+1}	$\leftarrow$	$n_1^k + 1$	n_1^{k+1}	$\leftarrow$	$n_1^k - 1$	n_1^{k+1}	$\leftarrow$	$n_1^k + 1$
n_0^{k+1}	$\leftarrow$	n_0^k	n_0^{k+1}	$\leftarrow$	$n_0^k + 1$	n_0^{k+1}	$\leftarrow$	n_0^k

At level k the partial FSTs all contain k Steiner points, $k > 1$. Let $n_0^k,\ n_1^k,\ n_2^k$ be the number of type zero, one, two Steiner points of a particular parent partial FST at level k. There are $2k + 1$ edges in the parent partial FST, from Table A.1 we have:

$$n_0^{k+1} = (2k+1)n_0^k + n_1^k \tag{A.2}$$

$$n_1^{k+1} = (2k+1)(n_1^k + 1) - 2n_1^k = (2k-1)n_1^k + 2k + 1 \tag{A.3}$$

$$n_2^{k+1} = (2k+1)n_2^k + n_1^k \tag{A.4}$$

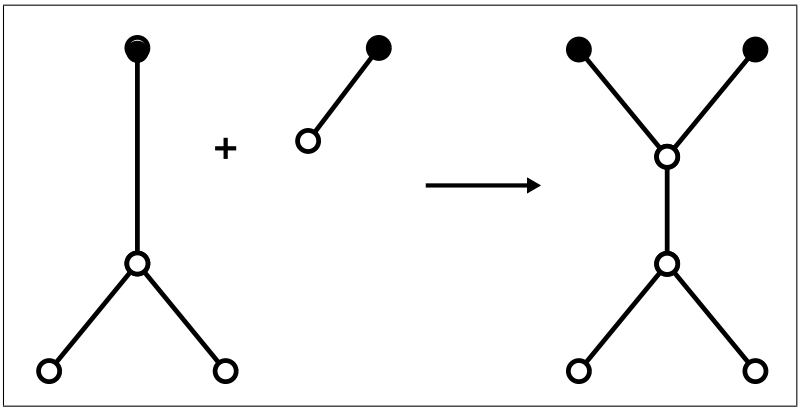


Figure A.2: Merging to a Steiner-terminal edge adjacent to a type one Steiner point increases the number of type two and type zero Steiner points by one and decreases the number of type one Steiner points by one. (Steiner points are shown in white, terminal nodes in black.)

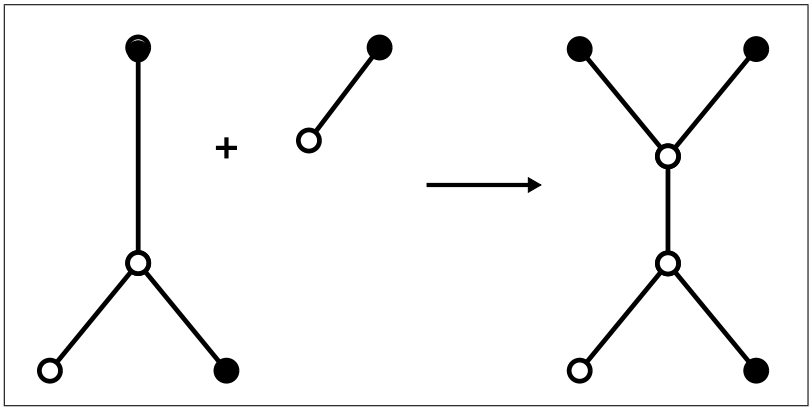


Figure A.3: Merging to a Steiner-terminal edge adjacent to a type two Steiner point increases the number of type one Steiner points by one. (Steiner points are shown in white, terminal nodes in black.)

A.2 The descendants over all parent topologies

In the previous section we computed the number of different type Steiner points at level $k+1$ *per* parent at level k. We now sum over all parents at level k to obtain the number of Steiner points of each type on level $k+1$.

Let N_2^k, N_1^k, and N_0^k denote the total number of type two, one and zero Steiner points among all partial FSTs at level k in Smith's enumeration scheme. Let $n_{2,j}^k$, $n_{1,j}^k$, and $n_{0,j}^k$ be the number of type two, one and zero Steiner points in topology j at level k. Letting T_k be the number of topologies at level k, then

$$N_i^k = \sum_{j=1}^{T_k} n_{i,j}^k \quad \text{for } i = 0, 1, 2.$$

We expand this sum for $i = 0, 1, 2$, using the identities established in equations (A.2), (A.3), and (A.4) for n_0^k, n_1^k, and n_2^k. For type two Steiner points ($i = 2$),

$$\begin{aligned} N_2^{k+1} &= \sum_{j=1}^{T_{k+1}} n_{2,j}^{k+1} \\ &= \sum_{j=1}^{T_k} (2k+1) n_{2,j}^k + n_{1,j}^k \\ &= (2k+1) \sum_{j=1}^{T_k} n_{2,j}^k + \sum_{j=1}^{T_k} n_{1,j}^k \\ &= (2k+1) N_2^k + N_1^k \end{aligned} \tag{A.5}$$

where the second equality uses (A.4). For type one Steiner points, ($i = 1$):

$$\begin{aligned} N_1^{k+1} &= \sum_{j=1}^{T_{k+1}} n_{1,j}^{k+1} \\ &= \sum_{j=1}^{T_k} \big((2k-1) n_{1,j}^k + 2k + 1 \big) \\ &= (2k-1) \sum_{j=1}^{T_k} n_{1,j}^k + \sum_{j=1}^{T_k} (2k+1) \\ &= (2k-1) N_1^k + T_k (2k+1) \\ &= \frac{(k+2)(k-2)}{2k-1} \end{aligned} \tag{A.6}$$

where the second equality uses (A.3). For type zero Steiner points, $(i = 0)$:

$$\begin{aligned} N_0^{k+1} &= \sum_{j=1}^{T_{k+1}} n_{0,j}^{k+1} \\ &= \sum_{j=1}^{T_k} \left((2k+1)n_{0,j}^k + n_{1,j}^k\right) \\ &= (2k+1)\sum_{j=1}^{T_k} n_{0,j}^k + \sum_{j=1}^{T_k} n_{1,j}^k \\ &= (2k+1)N_0^k + N_1^k \end{aligned} \tag{A.7}$$

where the second equality uses (A.2).

A.3 The number of different type Steiner points

Because T_k grows exponentially, it is convenient to normalize the values of N_i^k by dividing by T_k. Let $\tilde{N}_2^k$, $\tilde{N}_1^k$, and $\tilde{N}_0^k$ denote the average number of type two, one and zero Steiner points respectively *per topology* at level k. We have

$$\tilde{N}_i^k = \frac{N_i^k}{T_k} \quad \text{for } i = 0, 1, 2.$$

Further we note that T_{k+1} is the number of edges in the parent topologies in level k times the number of topologies in level k. That is,

$$T_{k+1} = (2k+1)T_k. \tag{A.8}$$

We also note that the number of terminal points at level k is the sum of N_1^k and $2N_2^K$ and is also the number of topologies T_k times the number of terminals per topology, $(k+2)$, resulting in the identity

$$N_1^k + 2N_2^k = T_k(k+2). \tag{A.9}$$

Finally, note that the number of Steiner points at level k is the sum of N_2^k, N_1^k, and N_0^K, and is also the number of topologies T_k times the number of Steiner points per topology, k, resulting in the identity

$$N_2^k + N_1^k + N_0^K = T_k k. \tag{A.10}$$

Dividing (A.5) by T_{k+1} we obtain a recursion for $\tilde{N}_2^k$:

$$\begin{aligned}
\tilde{N}_2^{k+1} &= \frac{(2k+1)N_2^k + N_1^k}{T_{k+1}} \\
&= \frac{N_2^k}{T_k} + \frac{N_1^k}{T_{k+1}} \\
&= \frac{N_2^k}{T_k} + \frac{T_k(k+2) - 2N_2^k}{T_{k+1}} \\
&= \frac{N_2^k}{T_k} + \frac{k+2}{2k+1} - 2\frac{N_2^k}{T_{k+1}} \\
&= \frac{N_2^k}{T_k} + \frac{k+2}{2k+1} - \frac{2}{2k+1}\frac{N_2^k}{T_k} \\
&= \tilde{N}_2^k - \frac{2}{2k+1}\tilde{N}_2^k + \frac{k+2}{2k+1} \\
&= \frac{2k-1}{2k+1}\tilde{N}_2^k + \frac{k+2}{2k+1}
\end{aligned}$$

where the second, fourth and fifth equalities use (A.8) and the third uses (A.9). The formula:

$$\tilde{N}_2^k = \frac{(k+1)(k+2)}{2(2k-1)} \tag{A.11}$$

satisfies the above recursion as follows,

$$\begin{aligned}
\tilde{N}_2^{k+1} &= \frac{(k+2)(k+3)}{2(2k+1)} \\
&= \frac{(k+1)(k+2) + 2k + 4}{2(2k+1)} \\
&= \frac{(k+1)(k+2)}{2(2k+1)} + \frac{k+2}{2k+1} \\
&= \frac{2k-1}{2k-1}\frac{(k+1)(k+2)}{2(2k+1)} + \frac{k+2}{2k+1} \\
&= \frac{2k-1}{2k+1}\frac{(k+1)(k+1)}{2(2k-1)} + \frac{k+2}{2k+1} \\
&= \frac{2k-1}{2k+1}\tilde{N}_2^k + \frac{k+2}{2k+1}.
\end{aligned}$$

Further, the closed form for $\tilde{N}_2^k$ satisfies $\tilde{N}_2^2 = 2$ in (A.11) and so is the correct formula for the average number of Steiner points incident to two terminal points at level k, $k \geq 2$.

It is now easy to find closed form expressions for the average number of the other type Steiner points at each level. Dividing the identity (A.9) by T_k, we obtain:

$$\begin{aligned}
\tilde{N}_1^k &= (k+2) - 2\tilde{(N)}_2^k \\
&= (k+2) - \frac{(k+1)(k+2)}{2k-1} \\
&= (k+2)\left(1 - \frac{k+1}{2k-1}\right) \\
&= (k+2)\left(\frac{k-2}{2k-1}\right).
\end{aligned}$$

Finally, dividing the the identity (A.10) by T_k we obtain:

$$\begin{aligned}
\tilde{N}_0^k &= k - \tilde{N}_1^k - \tilde{N}_2^k \\
&= k - \frac{(k+2)(k-2)}{2k-1} - \frac{(k+1)(k+2)}{2(2k-1)} \\
&= \frac{2k(2k-1) - 2(k+2)(k-2) - (k+1)(k+2)}{2(2k-1)} \\
&= \frac{(k-2)(k-3)}{2(2k-1)}.
\end{aligned}$$

Dividing the above expressions for $\tilde{N}_i^k$ by the number of Steiner points k per topology, we see the fraction of type two Steiner points is asymptotic to $1/4$. The derivative of the $\tilde{N}_2^k/k$ function is negative for $k > 1$, so the function is asymptotic *down* to $1/4$. The average fractions of type one and two Steiner points are asymptotic to $1/2$ and $1/4$, respectively. This indicates that on average more than one fourth of Steiner points are type two, so an exclusion criteria based on them could be useful. See Figure A.4 for the graph of the average fraction of different type Steiner points $\tilde{N}_i^k/k$, $i = 0, 1, 2$ as the number of Steiner points increases.

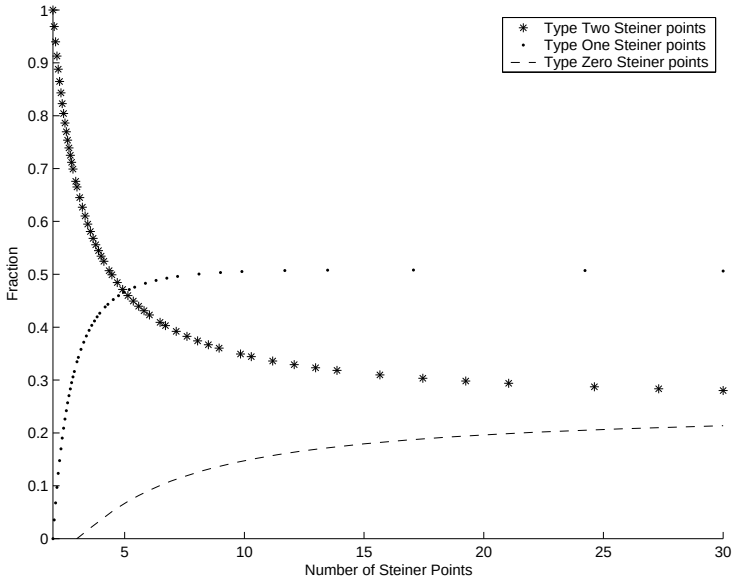


Figure A.4: The fraction of type two, type one, and type zero Steiner points as the number of Steiner points increases. Note that $\tilde{N}_2^k/k \searrow 0.25$.

REFERENCES

E.D Andersen and K.D Andersen. The MOSEK Optimization Software Version 6, 2010.

K.D. Andersen and E.D. Andersen. APOS user's manual for QMSN problems ver 1.71., 1996.

K.D. Andersen, E. Christiansen, A.R. Conn, and M.L. Overton. An efficient primal-dual interior-point method for minimizing a sum of Euclidean norms. *SIAM Journal on Scientific Computing*, 22(1):243–262, 2001.

S. Arora. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *Journal of the Association for Computing Machinery*, 45(5):753–782, 1998.

J. Beardwood, HJ Halton, and JM Hammersley. The shortest route through many points. In *Proc Camb Phil Soc*, volume 55, page 299, 1959.

J.E. Beasley. OR-Library: Distributing Test Problems by Electronic Mail. *The Journal of the Operational Research Society*, 41(11):1069–1072, 1990.

J.E. Beasley and F. Goffinet. A Delaunay triangulation-based heuristic for the Euclidean Steiner Problem. *Networks*, 24(4):215–224, 1994.

S.K. Chang. The Generation of Minimal Trees with a Steiner Topology. *Journal of the ACM (JACM)*, 19(4):699–711, 1972.

F.R.K. Chung and R.L. Graham. Steiner trees for ladders. *Annals of Discrete Mathematics*, 2:173–200, 1978.

D. Cieslik. The steiner ratio. *Combinatorial Optimization*, 10, 2001.

J. Clausen. Branch and bound algorithms-principles and examples. *Parallel Computing in Optimization*, pages 239–267, 1997.

EJ Cockayne. On Fermat's Problem on the Surface of a Sphere. *Mathematics Magazine*, 45(4):216–219, 1972.

R. Courant and H. Robbins. What is mathematics. *New York*, pages 199–201, 1941.

M. De Berg, O. Cheong, M. Van Kreveld, and M. Overmars. *Computational geometry: algorithms and applications*. Springer-Verlag New York Inc, 2008.

Carlos De la Mora. Personal correspondence, 2007.

J. Dolan, R. Weiss, and J. MacGregor Smith. Minimal length tree networks on the unit sphere. *Annals of Operations Research*, 33(7):501–535, 1991.

D.R. Dreyer and M.L. Overton. Two Heuristics for the Euclidean Steiner Tree Problem. *Journal of Global Optimization*, 13(1):95–106, 1998.

D. Du and X. Hu. *Steiner tree problems in computer communication networks*. World Scientific, 2008.

D.Z. Du and F.K. Hwang. A proof of the Gilbert-Pollak conjecture on the Steiner ratio. *Algorithmica*, 7(1):121–135, 1992.

D.Z. Du and W. Smith. Three disproofs of the Gilbert-Pollak conjecture on Steiner ratio in three or more dimensions. *J. Comhin. Thc. ory, Sw. A, in press. SJ Fortune. Voronoi diagrams and Delaunay triangulations. Coinpitting and Eiiclidcan GcwncJtry*, 1994.

M. Fampa and K.M. Anstreicher. An improved algorithm for computing Steiner minimal trees in Euclidean d-space. *Discrete Optimization*, 5(2):530–540, 2008.

MR Garey and DS Johnson. The rectilinear Steiner tree problem is NP-complete. *SIAM Journal on Applied Mathematics*, 32(4):826–834, 1977.

M.R. Garey, R.L. Graham, and D.S. Johnson. The complexity of computing Steiner minimal trees. *SIAM Journal of Applied Mathematics*, 32(4):835–859, 1977.

E.N. Gilbert and H.O. Pollak. Steiner Minimal Trees. *SIAM Journal on Applied Mathematics*, 16(1):1–29, 1968.

F. Glover. Tabu search: A tutorial. *Interfaces*, 20(4):74–94, 1990.

M. Hanan. On Steiner's problem with rectilinear distance. *SIAM Journal on Applied Mathematics*, 14(2):255–265, 1966.

FK Hwang. A linear time algorithm for full Steiner trees. *Operations Research Letters*, 4(5):235–237, 1986.

L. Ingber. Simulated annealing: Practice versus theory. *Mathematical and computer modelling*, 18(11):29–57, 1993.

R.M. Karp. Reducibility among combinatorial problems. *Complexity of computer computations*, 43:85–103, 1972.

J. Kleinberg and E. Tardos. *Algorithm Design*. Addison Wesley, 2005.

T. Koch and A. Martin. Solving Steiner tree problems in graphs to optimality. *Networks*, 32(3):207–232, 1998.

H.W. Kuhn. Steiner's problem revisited. *Studies in optimization*, 10:52–70, 1974.

P.J.M. Laarhoven and E.H.L. Aarts. *Simulated annealing: theory and applications*. Springer, 1987.

I. Ljubić, R. Weiskircher, U. Pferschy, G. Klau, P. Mutzel, and M. Fischetti. Solving the prize-collecting Steiner tree problem to optimality. In *Proceedings of ALENEX, Seventh Workshop on Algorithm Engineering and Experiments*. Citeseer, 2005.

M. Lundy. Applications of the annealing algorithm to combinatorial problems in statistics. *Biometrika*, 72(1):191, 1985.

Z.A. Melzak. On the problem of Steiner. *Canad. Math. Bull*, 4(2):143–148, 1961.

D.C. Montgomery. *Design and Analysis of Experiments*. John Wiley & Sons, New York, 5 edition, 2001.

Y. Nesterov and A. Nemirovsky. Interior point polynomial methods in convex programming. *Studies in applied mathematics*, 13, 1994.

Y.E. Nesterov and M.J. Todd. Primal-dual interior-point methods for self-scaled cones. *SIAM Journal on Optimization*, 8(2):324–364, 1998.

S.B. Rao and W.D. Smith. Improved approximation schemes for geometrical graphs via "spanners" and "banyans". In 30^{th} *ACM Symposium on Theory of Computing*, pages 540–550, 1998.

S. Ravada and A.T. Sherman. Experimental evaluation of a partitioning algorithm for the Steiner tree problem in R^2 and R^3 . *Networks*, 24(8):409–415, 1994.

J.M. Smith, D.T. Lee, and J.S. Liebman. An $O(N \log N)$ Heuristic for Steiner Minimal Tree Problems on the Euclidean Metric. *Networks*, 11:23–29, 1981.

J.M. Smith, R. Weiss, and M. Patel. An $O(N^2)$ heuristic for Steiner minimal trees in E^3. *Networks*, 26(4):273–289, 1995.

J.M.G. Smith and J.S. Liebman. Steiner trees, Steiner circuits and the interference problem in building design. *Engineering Optimization*, 4(1):15–36, 1979.

W.D. Smith. How to find Steiner minimal trees in Euclidean d-space. *Algorithmica*, 7(1):137–177, 1992.

J.F. Sturm and I. Polik. SeDuMi: Self Dual Minimization Version 1.1, 2006.

E.A. Thompson. The Method of Minimum Evolution. *Annals of Human Genetics*, 36(3):333–340, 1973.

B. Toppur and J.M.G. Smith. A Sausage Heuristic for Steiner Minimal Trees in Three-Dimensional Euclidean Space. *Journal of Mathematical Modelling and Algorithms*, 4(2):199–217, 2005.

D.M. Warme, P. Winter, and M. Zachariasen. GeoSteiner 3.1. Department of Computer Science, University of Copenhagen (DIKU), 2001.

P. Winter. An algorithm for the Steiner problem in the Euclidean plane. *Networks*, 15(3):323–345, 1985.

P. Winter and M. Zachariasen. Euclidean Steiner minimum trees: An improved exact algorithm. *Networks*, 30(3):149–166, 1997.

Guoliang Xue, Theodore P. Lillys, and David E. Dougherty. Computing the minimum cost pipe network interconnecting one sink and many sources. *SIAM J. on Optimization*, 10(1):22–42, 1999. ISSN 1052-6234. doi: http://dx.doi.org/10.1137/S1052623496313684.

M. Zachariasen. Local search for the Steiner tree problem in the Euclidean plane. *European Journal of Operational Research*, 119(2):282–300, 1999.

M. Zachariasen and P. Winter. Concatenation-Based Greedy Heuristics for the Euclidean Steiner Tree Problem. *Algorithmica*, 25(4):418–437, 1999.

L. Zhang. On the convergence of a modified algorithm for the spherical facility location problem. *Operations Research Letters*, 31(2):161–166, 2003.