J Heuristics (2011) 17:353–372
DOI 10.1007/s10732-010-9137-z

A randomized Delaunay triangulation heuristic for the Euclidean Steiner tree problem in $\Re^d$

Jon W. Van Laarhoven · Jeffrey W. Ohlmann

Received: 10 February 2008 / Revised: 5 March 2010 / Accepted: 15 June 2010 /
Published online: 9 July 2010

Abstract We present a heuristic for the Euclidean Steiner tree problem 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 identify the Steiner points to remove, and second-order cone programming to optimize the location of the remaining Steiner points. Unlike other ESTP heuristics relying upon Delaunay triangulation, we insert Steiner points probabilistically into Delaunay triangles to achieve different subtrees on subsets of terminal points. We govern this neighbor generation procedure with a local search framework that extends effectively into higher dimensions. We present computational results on benchmark test problems in $\Re^d$ for $2 \leq d \leq 5$.

Keywords Steiner tree · Delaunay triangulation

1 Introduction

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 so-called *Steiner points*) into the network to reduce its overall length. The ESTP is a difficult combinatorial optimization problem; Garey et al. (1977) show that recognition version of the ESTP is NP-hard, i.e., all problems in NP polynomially reduce

J.W. Van Laarhoven
Program in Applied Mathematics and Computational Sciences, University of Iowa,
14 MacLean Hall, Iowa City, IA 52242, USA
e-mail: jon-vanlaarhoven@uiowa.edu

J.W. Ohlmann (✉)
Department of Management Sciences, University of Iowa, 108 John Pappajohn Business Building,
Iowa City, IA 52242, USA
e-mail: jeffrey-ohlmann@uiowa.edu

to the ESTP, but the ESTP itself is not known to be in NP. The graph theory literature contains extensive studies of exact algorithms, approximation algorithms, and heuristic approaches for the ESTP.

Breakthroughs have lead to increases in the size of ESTP instances which can be solved to optimality by exact algorithms, particularly in $\Re^2$. The Geosteiner algorithm (Warme et al. 2001), can optimally solve ESTP instances with up to 1000 terminal points. For higher dimensions, however, exact algorithms are quite limited as many of the pruning rules are highly specialized to the planar case. 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 to locate the Steiner points and "strong branching" to accelerate the fathoming process in the branch-and-bound enumeration scheme. The Smith+ algorithm is capable of optimally solving instances with up to 16 terminal points in $\Re^d$ for $2 \le d \le 5$.

Arora (1998) shows ESTP belongs to the class of NP-hard problems which have a polynomial-time approximation scheme (PTAS). Using recursive partitioning 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) describe a $(1 + 1/c)$-PTAS which uses a generalization of graph "spanners" and runs in $2^{(cd)^{\mathcal{O}(d)}}n + (cd)^{\mathcal{O}(d)}n \log n$ time on instances with n terminal points in $\Re^d$. Practical implementation of these polynomial time approximations remains to be shown.

In this paper, we focus on a heuristic approach to the ESTP that utilizes computational geometry and randomization to obtain near-optimal solutions for instances in $\Re^d$, $2 \le d \le 5$, within reasonable computation times. The remainder of the paper is organized as follows. We present relevant terminology and background information in Sect. 2. We outline our algorithm in Sect. 3 and compare its features to existing heuristics in Sect. 4. In Sect. 5, we analyze the computational performance of our heuristic against both randomized and highly structured benchmark instances from the literature. We offer concluding remarks in Sect. 6.

2 Background information

In this section, we briefly review research on heuristics which contain features relevant to our work. For a comprehensive review of planar ESTP heuristics, see Zachariasen (1999).

An optimal solution to an ESTP instance is called a *Steiner minimal tree* (SMT). The SMT is closely related to the well-known minimum spanning tree (MST), which is the shortest network connecting a set of terminal points, X, without the addition of extra (Steiner) points ($S = \emptyset$). The MST for a graph of n points can be computed in $\mathcal{O}(n \log n)$ time (Preparata and Shamos 1988). In $\Re^2$, the ratio of the length of the SMT and the length of the MST, is always greater than or equal to $\sqrt{3}/2 \approx 0.866$ (Du and Hwang 1992), and the minimum *Steiner ratio* is achieved for a network of terminal points located at the vertices of an equilateral triangle. 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.

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 the set of Steiner points. This relationship between the MST and SMT motivates many ESTP heuristics which attempt to identify "good" candidate Steiner points 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 node 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°. Gilbert and Pollak (1968) provide mathematical proofs of these and other facts regarding Steiner trees.

In one of the first heuristic approaches for the planar ESTP, Thompson (1973) describes a constructive approach 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. At each iteration, a single Steiner point is inserted into $\triangle uvw$ 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. An insertion is performed 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 latter step can be viewed as an early relatively minimal tree (RMT) algorithm; RMT algorithms solve for the optimal locations of Steiner points for a prespecified topology. 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 the terminal points and determining the optimal location of the Steiner points after all insertions have been executed.

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 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 Fig. 2, but is unable to detect the Steiner point needed in the middle for the optimal solution as shown in Fig. 3.

Beasley and Goffinet (1994) present an algorithm which generates candidate Steiner points for the planar ESTP using computational geometry. Specifically, Beasley and Goffinet (1994) iteratively apply the *Delaunay triangulation* to $X \cup S$. 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

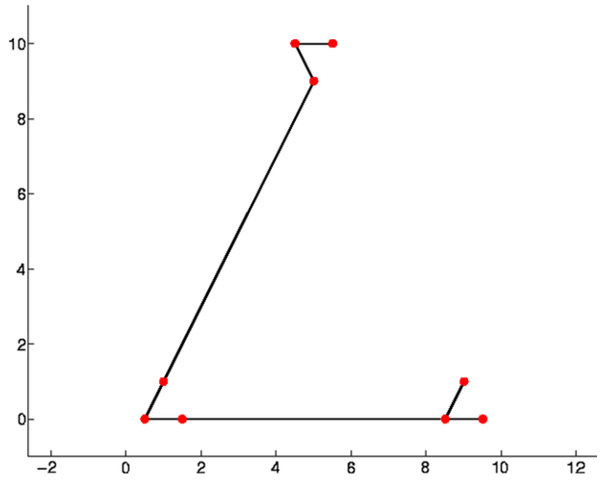

Fig. 1 The MST for the nine-point instance, which provides the initial solution for Steiner insertion heuristics

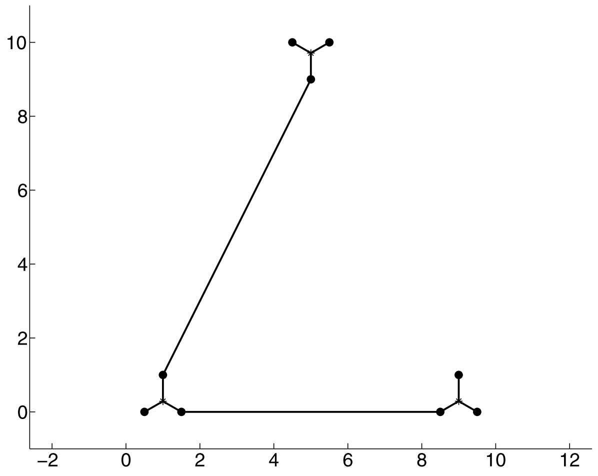

Fig. 2 Edge insertion achieves a sub-optimal solution to the nine-point instance

in $\mathrm{DT}(Z)$ 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 $\mathrm{DT}(Z)$.

The Delaunay triangulation is the straight-line dual graph of the *Voronoi diagram*. The Voronoi diagram of Z, a set of points in $\Re^d$, is determined by partitioning the Euclidean space into convex polyhedra (called *Voronoi regions*) with one node in each region. For every point $z \in Z$, the Voronoi region about z denoted $\mathrm{vor}(z)$ consists of all points closer to z than to any other point in Z. That is $\mathrm{vor}(z) = \{u : \|u - z\| < \|u - y\|\ \forall y \in Z\}$.

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

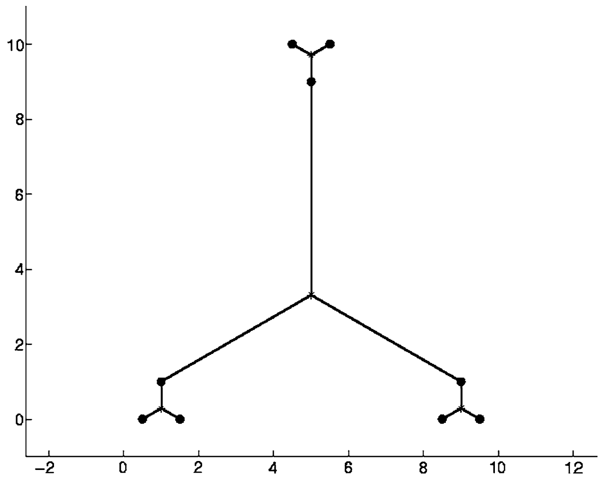

Fig. 3 The optimal solution for the nine-point instance contains a centrally-located Steiner point

plane occurs in an equilateral triangle. In $\Re^3$, a large reduction is achieve for the regular tetrahedral simplex and Toppur and Smith (2005) demonstrate that by linking many regular tetrahedral simplices together, even larger reductions can be obtained. A second property is that the edges of the Delaunay triangulation of a set of points, Z, contain the MST of Z. 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. However, as Figs. 1, 2, and 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, 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). In Fig. 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.

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 $\mathrm{DT}(X \cup S)$ 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

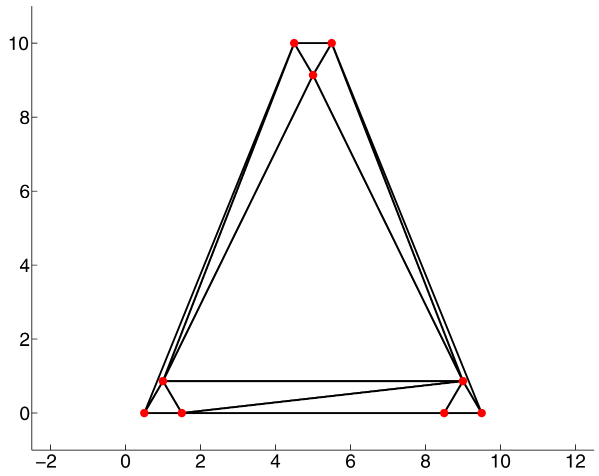

Fig. 4 The Delaunay triangulation for the nine-point instance identifies the central Steiner point location requisite for the SMT

condition. A simulated annealing framework governs the evaluation of the neighbor solution generated by this process and upon rejection or acceptance of the neighbor solution, the algorithm returns to the expansion phase.

Another well-known SMT property is that a SMT of X is a concatenation of *full Steiner trees* on subsets of X, where a full Steiner tree (FST) is a network on n terminal points which contains $(n-2)$ Steiner points. For example, the SMT in Fig. 3 is a concatenation of four FSTs, each of three terminal points. This observation forms the basis of the Geosteiner algorithm (Warme et al. 2001), which builds a list of all candidate FSTs which could appear in the SMT and executes a branch-and-bound procedure to concatenate elements of the candidate list into the optimal SMT. Zachariasen and Winter (1999) present a heuristic approach for the planar ESTP which utilizes geometric structures (Gabriel graphs, Delaunay triangles, minimal spanning trees, etc.) to construct a restricted candidate list F of FSTs on subsets of terminal points. The FSTs are then greedily concatenated into a candidate Steiner tree.

Relative to the planar ESTP, the literature on heuristics for the ESTP in $\Re^d$ for $d > 2$ is limited. Smith et al. (1995) utilize the Delaunay triangulation and MST to decompose specially-structured sausage-shaped instances into smaller subproblems which are then solved by an exact algorithm. Toppur and Smith (2005) extend the heuristic to random instances in $\Re^3$ using decomposition by Delaunay tetrahedrons. 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).

3 Algorithm description

In this section, we describe our heuristic. The algorithm utilizes 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)

Algorithm 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 $T = \text{MST}(X)$.
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 $T' = \text{MST}(X \cup S')$.
    $T' \leftarrow$ Clean$(T')$ via Algorithm 2.
    $T' \leftarrow$ EdgeInsertion$(T')$ via Algorithm 3.
    if $\ell(T') - \ell(T) < \chi \ell(T)$ then
        $T' \leftarrow$ Recover$(T')$ via Algorithm 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
```

as an attempt to correct any violations of the angle condition. We govern this neighbor generation procedure with a first-improvement local search framework, which we outline in Algorithm 1.

Recall that X denotes the set of terminal points in Euclidean space, and S is the set of Steiner points. We initialize $S = \emptyset$, and begin the algorithm by constructing the Delaunay triangulation of $X \cup S$. We sequentially consider each Delaunay simplex, inserting a Steiner point at the simplex centroid with probability p; 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 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 (treating its neighbors as fixed); 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. As an intermediate check on the validity of the topology, we form the MST on the terminal points and current

Algorithm 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 $T'' = \mathrm{MST}(X \cup S'')$
until each Steiner point $s \in S''$ has degree $\geq 3$
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
```

set of Steiner points. 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 optimize the location of the Steiner points using the second-order cone solver of SeDuMi (Sturm and Polik 2006) if Steiner point to Steiner point edges exist in the tree. In such a case, the previous local optimization of Steiner points (treating neighbor points as fixed) may not have resulted in the globally optimal location for all Steiner points in the tree. For details regarding formulating the problem of locating the Steiner points for a given full Steiner topology as a conic optimization problem, we refer to Fampa and Anstreicher (2008). 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 the degenerate Steiner point and appropriately reattach the tree.

Algorithm 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 $i = 1, \dots, |X \cup S|$ 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 $120^\circ$ 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).
```

After completing the cleanup operations of Algorithm 2, we perform an edge insertion as outlined in Algorithm 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. 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 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.

4 Discussion of algorithm features

Our algorithm maintains many features of other ESTP heuristics relying upon 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.

Algorithm 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 $i = 1, \dots, |X \cup S|$ 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 $j = 1, \dots,$ 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).
```

1981). The difference in insertion strategy allows our algorithm to achieve different FSTs on subsets of terminal points.

Figures 5, 6, 7, and 8 demonstrate how our probabilistic Steiner insertion can quickly achieve FSTs that a deterministic procedure may not. Figure 5 illustrates the minimal spanning tree and corresponding Delaunay triangulation for the sixth instance of size-10 ESTP instances from the OR-Library (Beasley 1990). The approaches of Smith et al. (1981) and Beasley and Goffinet (1994) struggle to achieve a SMT for This is an instance for which t. Our local search heuristic finds the optimal solution in Fig. 6. The key region is the two lower left-hand Delaunay triangles. Figure 7 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 Fig. 6. 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, Fig. 8 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 Fig. 6.

While Beasley and Goffinet (1994) do not consider randomization in the generation of their candidate solutions, they accept non-improving solutions probabilisti-

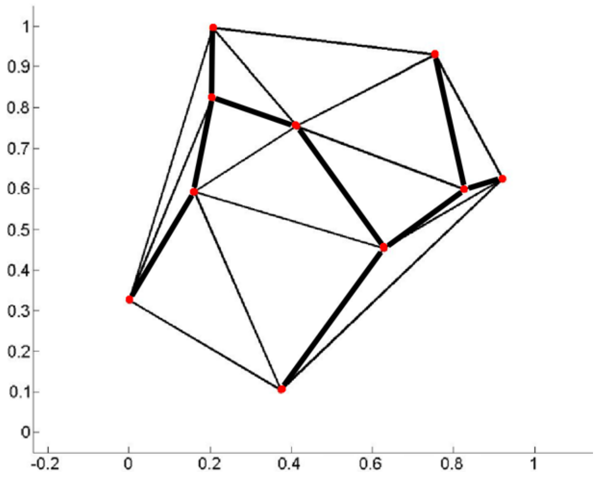

Fig. 5 As demonstrated by the sixth instance of the size-10 ESTP instances from the OR-Library, the Delaunay triangulation contains the edges of the MST (*highlighted in bold*)

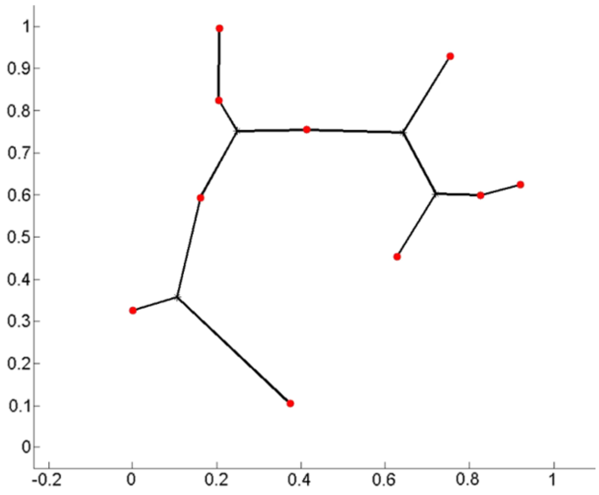

Fig. 6 A SMT for the sixth instance of the size-10 ESTP instances from the OR-Library contains a FST on three terminal points in the lower left-hand region of the network

cally 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 our heuristic generates neighbor solutions probabilistically and accepts them deterministically. As our computational results in Sect. 5 attest, our 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 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

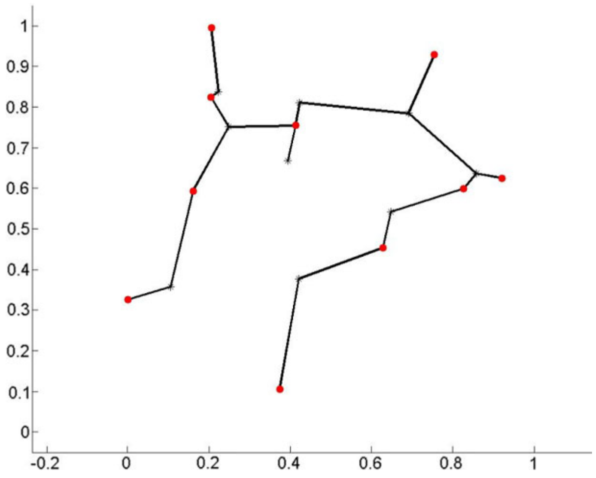

Fig. 7 Deterministic Steiner point insertion in all nine non-degenerate Delaunay triangles results in a pair of degree-two Steiner points in the lower left-hand region which will be removed by a cleanup operation

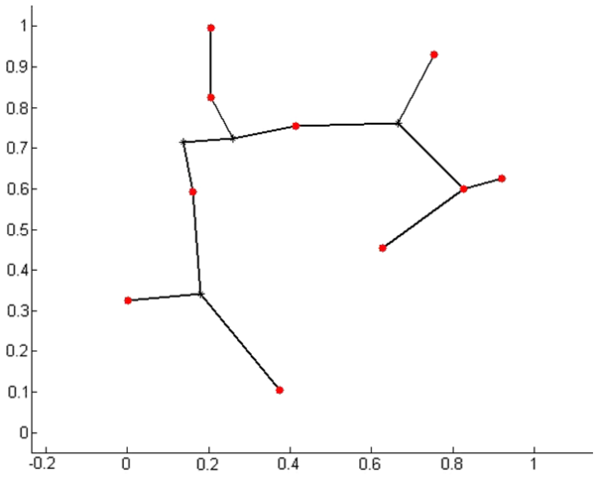

Fig. 8 Probabilistic Steiner point insertion identifies the lower left FST in the SMT

which requires $d - 1$ Steiner points to form a FST. We address this issue by simply inserting a single point at the simplex centroid and allow later iterations of the algorithm to insert additional Steiner points in this simplex.

Our post-processing procedures applied to the candidate tree after the trial insertion of Steiner points are simplified 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, our cleanup procedure (Algorithm 2) addresses only Steiner points of less than degree three and attempts to address the remaining untenable Steiner points by inserting edges via Algorithm 3. Our application of Algorithms 2 and 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 to directly address points with degree larger than three only in candidate trees within χ percent of the current solution. Our computational work, e.g., Table 6 in Sect. 5, suggests that addressing high-degree points via Algorithm 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.

5 Computational results

We implement our algorithm in Matlab and execute the computational experiments on dual core 2.4 GHz Pentium processor with 2 GB of RAM and a Windows operating system. We compare our approach to Beasley and Goffinet (1994) and Zachariasen (1999) on 195 planar ESTP instances from the OR-Library (Beasley 1990). We also test our algorithm on 20 random instances in $\Re^d$, $3 \le d \le 5$ 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, we define $\sigma = 100 * (\ell(\text{MST}) - \ell(T))/\ell(\text{MST})$ as the percent reduction over the MST. The ratio $\rho = \ell(T)/\ell(\text{MST})$ is also commonly used as a measure of quality, and we adopt this 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 our algorithm twenty times on each instance and report statistics regarding the best solution, average solution, and performance variability. Tables 2 and 3 present results by aggregating individual instances by problem size. Each entry in Table 2 is computed over 15 instances. Each entry of Table 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 $\sigma_{i,best}^n = \max\{\sigma_{i1}^n, \dots, \sigma_{i,20}^n\}$ and let the average percent reduction be $\bar{\sigma}_{i\cdot}^n = (\sigma_{i1}^n + \cdots + \sigma_{i,20}^n)/20$. For a problem class of m instances of size n, we report the overall best percent reduction as $\sigma_{best}^n = (\sigma_{1,best}^n + \cdots + \sigma_{m,best}^n)/m$ and the overall average percent reduction as $\bar{\sigma}^n = (\bar{\sigma}_{1\cdot}^n + \cdots + \bar{\sigma}_{m\cdot}^n)/m$. We measure the variability of solution quality for the aggregated results 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 the 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 and 5 in terms of ρ. Specifically, $\rho_{best} = \min\{\rho_1, \dots, \rho_{20}\}$ represents the best solution obtained out of twenty runs on a specified instance. Similarly, average solution quality is given by $\bar{\rho} = (\rho_1 + \cdots + \rho_{20})/20$ and performance variability is given by the standard deviation of $\rho_1, \dots, \rho_{20}$.

Table 1 Insertion probabilities between 0.3 and 0.6 consistently achieve the largest average percent reduction for the OR-Library ESTP instances (Beasley 1990)

	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

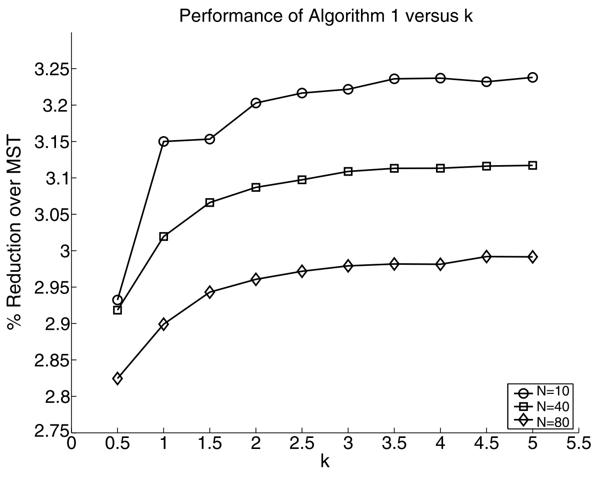

Fig. 9 For instances of size 10, 40, and 80 from the OR-Library (Beasley 1990), this graph of average percent reduction versus k, where trial limit $= k\sqrt{n}$, demonstrates that a value of $k = 3$ results in an iteration count that provides an acceptable tradeoff between solution quality and solution time

To demonstrate the role of the insertion probability, p, we perform computational experiments in which we fix p at various values ranging from 0 to 1 and execute Algorithm 1 on the planar ESTP instances from the OR-Library (Beasley 1990). Each entry of Table 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.

To implement our algorithm, we must establish a value for trial limit to control the loop iterations for the local search. Figure 9 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.

Table 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 2 shows, our 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 our 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 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 $[0, 10]^d$ hypercube. As Table 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 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 our 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, and provide the optimal solutions. As Table 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 general randomized Delaunay triangulation approach is very competitive.

Table 2 Our heuristic is competitive with other approaches on the planar OR-Library instances (Beasley 1990)

n	Beasley and Goffinet (1994)		Zachariasen and Winter (1999)		Randomized Delaunay triangulation			Opt. %
	Avg. % reduction	CPU (min)	Avg. % reduction	CPU (sec)	σ^n_{best}	$\bar{\sigma}^n \pm \sqrt{SS_E/(9m)}$	CPU (sec)	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

Table 3 Our heuristic achieves results comparable with the exact approach of Fampa and Anstreicher (2008) with a fraction of the computational effort

d	n	Smith + algorithm		Randomized Delaunay triangulation			
		Opt. % reduction	CPU (sec)	σ^n_{best}	$\bar{\sigma}^n \pm \sqrt{SS_E}$	Optimal runs	CPU (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 Our heuristic consistently obtains optimal or near-optimal solutions on the $\Re$-sausage instances in $\Re^3$

n	Toppur and Smith (2005)	Randomized Delaunay triangulation		
	$\rho^\star$	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

Table 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.

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 5 attests that our approach performs well as the average solution obtained is never more than seven percent above this lower bound.

Table 5 Performance comparison on $\Re^3$ instances appearing in Smith et al. (1995) and Toppur and Smith (2005)

Instance	n	Toppur and Smith (2005)	Smith et al. (1995)	Randomized Delaunay triangulation		
		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 6 We report the percentage of improving candidate trees which require a Steiner topology to be recovered 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

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). For the implemented value of $\chi = 0$, we report in Table 6 the percentage of improving candidate trees that require Algorithm 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, where percent improvement for a modification of tree T to tree T' is defined by $(\ell(T) - \ell(T'))/\ell(T)$.

Table 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 improves these trees by less than 0.5 percent. For the higher-dimension instances of Fampa and Anstreicher (2008), Algorithm 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), Algorithm 4 is required much 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 to attain a Steiner topology for a candidate tree in a non-planar instance, but there does not appear to be much opportunity for improving the candidate trees with non-Steiner topologies in any dimension.

6 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.

Acknowledgements The authors would like to thank the two anonymous referees for their constructive comments that lead to the improvement of this paper.

References

Arora, S.: Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. J. Assoc. Comput. Mach. **45**(5), 753–782 (1998)
Beasley, J.E.: OR-Library: Distributing test problems by electronic mail. J. Oper. Res. Soc. **41**(11), 1069–1072 (1990)
Beasley, J.E., Goffinet, F.: A Delaunay triangulation-based heuristic for the Euclidean Steiner problem. Networks **24**(4), 215–224 (1994)
Chung, F.R.K., Graham, R.L.: Steiner trees for ladders. Ann. Discrete Math. **2**, 173–200 (1978)
Dreyer, D.R., Overton, M.L.: Two heuristics for the Euclidean Steiner tree problem. J. Glob. Optim. **13**(1), 95–106 (1998)
Du, D.Z., Hwang, F.K.: A proof of the Gilbert-Pollak conjecture on the Steiner ratio. Algorithmica **7**(1), 121–135 (1992)
Fampa, M., Anstreicher, K.M.: An improved algorithm for computing Steiner minimal trees in Euclidean d-space. Discrete Optim. **5**(2), 530–540 (2008)
Garey, M.R., Graham, R.L., Johnson, D.S.: The complexity of computing Steiner minimal trees. SIAM J. Appl. Math. **32**(4), 835–859 (1977)
Gilbert, E.N., Pollak, H.O.: Steiner minimal trees. SIAM J. Appl. Math. **16**(1), 1–29 (1968)
Montgomery, D.C.: Design and Analysis of Experiments, 5th edn. Wiley, New York (2001)
Preparata, F.P., Shamos, M.I.: Computational Geometry: An Introduction, 2nd edn. Springer, New York (1988)
Rao, S.B., Smith, W.D.: Improved approximation schemes for geometrical graphs via “spanners” and “banyans”. In 30th ACM Symposium on Theory of Computing, pp. 540–550 (1998)
Ravada, S., Sherman, A.T.: Experimental evaluation of a partitioning algorithm for the Steiner tree problem in R^2 and R^3. Networks **24**(8), 409–415 (1994)
Smith, W.D.: How to find Steiner minimal trees in Euclidean d-space. Algorithmica **7**(1), 137–177 (1992)
Smith, J.M., Lee, D.T., Liebman, J.S.: An $O(N \log N)$ heuristic for Steiner minimal tree problems on the Euclidean metric. Networks **11**, 23–29 (1981)
Smith, J.M., Weiss, R., Patel, M.: An $O(N^2)$ heuristic for Steiner minimal trees in E^3. Networks **26**(4), 273–289 (1995)
Sturm, J.F., Polik, I.: SeDuMi: self dual minimization version 1.1 (2006)
Thompson, E.A.: The method of minimum evolution, Ann. Hum. Genet. **36**(3), 333–340 (1973)
Toppur, B., Smith, J.M.G.: A sausage heuristic for Steiner minimal trees in three-dimensional Euclidean space. J. Math. Model. Algorithms **4**(2), 199–217 (2005)
Warme, D.M., Winter, P., Zachariasen, M.: GeoSteiner 3.1. Department of Computer Science, University of Copenhagen (DIKU) (2001)
Zachariasen, M.: Local search for the Steiner tree problem in the Euclidean plane. Eur. J. Oper. Res. **119**(2), 282–300 (1999)
Zachariasen, M., Winter, P.: Concatenation-based greedy heuristics for the Euclidean Steiner tree problem. Algorithmica **25**(4), 418–437 (1999)