Steiner Tree Heuristics in Euclidean *d*-Space

Andreas E. Olsen[1], Stephan S. Lorenzen[1], Rasmus Fonseca[1], and Pawel Winter[1]

Department of Computer Science, University of Copenhagen,
Universitetsparken 5, 2100 Copenhagen O, Denmark
mr.foxhide@hotmail.com, stephan.lorenzen@gmail.com,
{rfonseca,pawel}@di.ku.dk

Abstract. We present a class of heuristics for the Euclidean Steiner tree problem in a d-dimensional space, $d \geq 3$. These heuristics identify small subsets with few, geometrically close, terminals using Delaunay tessellations and minimum spanning trees. Low cost spanning trees of these subsets are determined by applying the exact algorithm for the Euclidean Steiner minimal tree in d-space as well as its heuristic modifications. These low cost spanning trees are sorted according to an appropriately chosen measure of quality. A tree spanning all terminals is constructed using greedy concatenation. Computational experiments indicate that problem instances with 80,000 terminals in R^3, 41,000 terminals in R^4, 15,000 terminals in R^5, and 3,800 terminals in R^6, can be solved within 1 minute while producing high quality solutions.

Keywords: Euclidean Steiner tree, heuristic, d-dimensional space

1 Introduction

Let R^d denote the d-dimensional Euclidean space, $d \geq 2$. The coordinates of a point $p \in R^d$ are specified by d numbers, i.e., $p = (p_1, p_2, \ldots, p_d)$. The length of a line segment $e = (u, v)$ between two points u and v is given by

$$||e|| = \sqrt{\sum_{i=1}^{d} (v_i - u_i)^2} \tag{1}$$

Let $T = \{t_1, t_2, \ldots, t_n\}$ denote a set of n *terminals* in R^d. The *Euclidean Steiner minimal tree* (SMT) problem for T in R^d asks for the shortest connected network $N = (V, E)$, $T \subseteq V$. The length $||N||$ of N is the sum of lengths of its line segments. N has to be a tree. If $V = T$, then N is the Euclidean minimum spanning tree (MST) of T. It usually is not the shortest network spanning T. If line segments are permitted to meet at T and at appropriately located *Steiner points* $S = V \setminus T$, a shorter *Steiner* tree spanning T can be obtained. Consider for example 3 terminals placed at the corners of a unit length equilateral triangle. Its MST, shown in Fig. 1 (left), has length 2 while the shortest possible network, shown in Fig. 1 (right), has length $\sqrt{3}/2$.

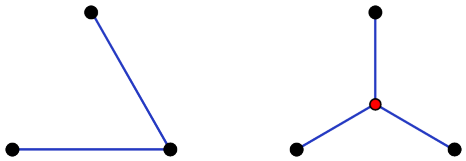

Fig. 1. MST and SMT for 3 corners of an equilateral triangle.

The Euclidean Steiner tree problem in R^2 has been extensively studied in the literature. GeoSteiner [18, 11] is an exact algorithm that can solve to optimality fairly large problem instances with thousands of terminals. Also efficient and practical heuristics have been suggested [19]. The same problem but in R^d, $d \geq 3$, has received considerable less attention. Furthermore, it seems to be much more difficult. Exact algorithms [15, 6, 7] can approximate optimal solutions for not more than 12-18 terminals. Very few heuristic solutions (focusing on the R^3 case) have been suggested in the literature [14, 17]. The purpose of this paper is to generalize one of the most efficient heuristics in R^2 [13, 19] to higher dimension spaces, and to provide computational results of its performance.

The paper is organized as follows. Section 2 gives the definitions, discusses some basic properties of the Euclidean SMTs and mentions some Steiner ratio results. The numerical optimization algorithm that can approximate the Euclidean SMT is discussed in Section 3. The heuristic (that uses this numerical optimization algorithm to solve small problem instances) is described in Section 4. Section 5 discusses three different ways of finding good solutions to small problem instances (with up to 7 terminals). Section 6 discusses how to identify small subsets of terminals with good Steiner ratios as well as methods of finding good solutions to such subsets. Concatenation of low cost trees for covered d-simplices and sausages to obtain a low cost tree for all terminals is discussed in Section 7. Computational results are given in Section 8 while conclusions are given in Section 9.

2 Definitions

A *k-simplex* in R^d is a polytope which is the convex hull of $k+1$ affinely independent points, $k \leq d$. The convex hull of any nonempty proper subset of extreme points of a k-simplex is called a *face* of the simplex. A *facet* of a k-simplex is any face which is also a $(k-1)$-simplex. A *simplicial complex* $\mathcal{K}$ is a set of simplices that satisfy the following conditions:

- Any face of a simplex from $\mathcal{K}$ is also in $\mathcal{K}$.
- The intersection of any two simplices $\sigma_1, \sigma_2 \in \mathcal{K}$ is a face of both σ_1 and σ_2.

A Delaunay tessellation $DT(T)$ of T in R^d is a maximal collection of d-simplices such that no terminal of T is inside the d-sphere circumscribing a

d-simplex in $DT(T)$. $DT(T)$ is a simplicial complex. Let $MST(T)$ denote the Euclidean MST of T. It is well-known that $MST(T)$ is a subgraph of $DT(T)$.

Given a pair of d-simplices σ_1 and σ_2 in $DT(T)$ that share a facet, their union is a simplicial complex consisting of $d+2$ terminals from T. It is denoted by $\sigma_1 \oplus \sigma_2$. The union operator extends in a natural way to larger facet-sharing simplicial complexes.

A *d-sausage* S_m, $m \geq 1$, in $DT(T)$ is defined recursively as follows:

- S_1 is a d-simplex in $DT(T)$. Assume that its $d+1$ terminals have distinct consecutive integer labels.
- Let σ be a d-simplex in $DT(T)$ consisting of the d highest (or lowest) labeled terminals of S_m and a terminal $t \notin S_m$. Give t the label one greater than (or one less than) any label in S_m and let $S_{m+1} = S_m \oplus \sigma$.

A d-sausage is *regular* if it consists of regular d-simplices. A regular 3-sausage is shown in Fig. 2.

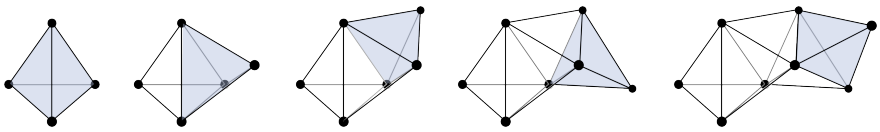

Fig. 2. 3-sausages are formed using the corners of a sequence of regular tetrahedra sharing faces

2.1 Properties

Let $SMT(T)$ denote the Euclidean SMT for a set of terminals T in R^d. Let E denote the line segments (edges) of $SMT(T)$ and let S denote Steiner points of $SMT(T)$. $SMT(T)$ has, among others, the following properties [10]:

- It is a tree.
- Edges in E are non-intersecting line segments.
- Every $t \in T$ has degree at most 3. Every $s \in S$ has degree 3.
- Line segments meeting at a terminal make at least 120^o angles with each other. Line segments meeting at a Steiner point make exactly 120^o angles with each other.
- $|S| \leq n-2$ and $|E| \leq 2n-3$.

Any tree with the above properties is called a *Steiner tree* of T. If it has $n-2$ Steiner points then it is called a *full Steiner tree* of T. If the locations of Steiner points are not important, a *(full) Steiner topology* (FST) defines the connections between terminals and Steiner points. The exact algorithm for the Euclidean SMT problem in R^2 [18] finds full Steiner trees for well-chosen (usually few)

FSTs. Among all possible ways of concatenating these full Steiner trees, a union minimizing the total weight and spanning all terminals yields the $SMT(T)$.

Smith's algorithm for the Euclidean SMT in R^d, $d \geq 2$, works in a different way [15]. It enumerates all FSTs of T. For a given FST, a numerical method is used to approximate the shortest tree with this FST. It exists, it is unique, and it is called a *relatively minimal tree* (RMT) for this FST. The shortest among all RMTs yields $SMT(T)$. RMTs may be *degenerate* if they have zero length line segments. This can happen when Steiner points overlap with terminals or with each other. In the latter case, such an RMT cannot be the $SMT(T)$.

It can be shown [10] that the number $f(n)$ of FSTs for a set of n terminals, $n \geq 3$, is given by

$$f(n) = 1 \times 3 \times 5 \times \ldots \times (2n-5) = \frac{(2n-4)!}{(n-2)!2^{n-2}} \tag{2}$$

2.2 Steiner Ratio

Let

$$\rho_d(T) = \frac{||SMT(T)||}{||MST(T)||} \tag{3}$$

where T is a d-dimensional set of terminals. Clearly $\rho_d(T) \leq 1$. Define the *Steiner ratio*

$$\rho_d = \inf_T \{\rho_d(T)\} \tag{4}$$

It has been conjectured [9] that

$$\rho_2 = \frac{\sqrt{3}}{2} \approx 0.866025\ldots \tag{5}$$

While this conjecture is believed to be true, its proof has not been obtained yet. It has also been conjectured [9] that ρ_d, $d \geq 2$, is achieved when T consists of extreme points of a regular d-simplex. For example, it was conjuctured that

$$\rho_3 = \frac{1+\sqrt{6}}{3\sqrt{2}} \approx 0.813052\ldots \tag{6}$$

However, this conjecture was disproved for $3 \leq d \leq 9$ [15] and subsequently for any $d \geq 3$ [4]. It has been shown [16] that the Steiner ratio for the regular 3-sausage S_m is 0.8080649361 when $m = 3$. It decreases as m increases and it is bounded from above by 0.7841903733771 as $m \to \infty$.

3 Approximation Scheme for the SMT Problem

The Euclidean SMT problem in R^d is NP-hard even for $d = 2$ [8]. It also has been shown that a geometrical construction approach for $d \geq 3$ would require solving eight-degree polynomials [15]. As a consequence, numerical approaches are the

only way to approximate optimal solutions of the Euclidean SMT problem in R^d, $d \geq 3$. The problem remains unsolvable by exact methods for $d \geq 3$ even if $n = 4$, or if the FST of $SMT(T)$ is known.

The approximation scheme for the Euclidean SMT problem in R^d, $d \geq 2$, already briefly mentioned in Subsection 2.1, is quite slow and practical only for $n \leq 12$ [15]. Improved versions [6, 7] are still slow but work for $n \leq 18$. Since the approximation scheme and (more importantly) heuristics based on the approximation scheme will be used to find good solutions for small subsets of terminals, a brief description is given here.

The approximation scheme generates all FSTs with n terminals. For each FST, it approximates its unique, possibly degenerate, RMT.

3.1 Generation of FSTs

There is only one FST for T_3. Assume that terminals of T are labeled $\{t_1, t_2, ..., t_k\}$. Let $T_k = \{t_1, t_2, ..., t_k\}$ for any k, $3 \leq k \leq n$. FSTs of T_k, $3 < k \leq n$, are constructed from FSTs of T_{k-1}. Let $\mathcal{T}_{k-1}$ denote one of the $f(k-1)$ FSTs of T_{k-1}. It has $k-3$ Steiner points $s_{n+1}, s_{n+2}, \ldots, s_{n+k-3}$ and $2k-5$ edges. Let s_{n+k-2} denote the Steiner point incident with t_k in every FST of T_k. Create $2k-5$ FSTs of T_k by inserting s_{n+k-2} into every edge of $\mathcal{T}_{k-1}$. By applying this *expansion* procedure to every FST of T_{k-1}, every FST of T_k is generated exactly once. An example showing the generation of 3 FSTs of T_4 by expanding the unique FST of T_3 is shown in Fig. 3.

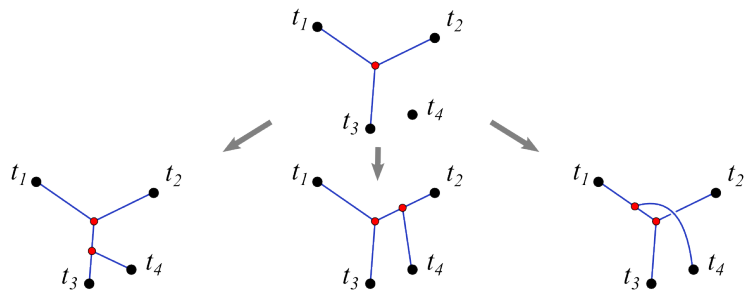

Fig. 3. FSTs of T_4 obtained by connecting t_4 to either of the edges from the unique FST of T_3.

3.2 Numerical Optimization for a Given FST

Given an FST $\mathcal{T}_n$, arbitrary initial positions are assigned to its $n-2$ Steiner points. The positions of Steiner points are recomputed iteratively so that the total length of the tree is reduced until an appropriately chosen threshold is

reached. In order to be able to distinguish between locations of Steiner points in consecutive iterations, let v_j^i denote the location of t_j, $j = 1, 2, \ldots, n$, and s_j, $j = n+1, n+2, \ldots, 2n-2$, before the i-th iteration, $i \geq 1$. Note that $v_j^{(i)} = v_j^{(i+1)}$ for all $j = 1, 2, \ldots, n$, and for all $i \geq 1$.

Consider the system of $n-2$ equations with $n-2$ unknown (corresponding to the locations of Steiner points)

$$\sum_{v_j^i v_l^i \in \mathcal{T}_n} \frac{v_j^{i+1} - v_l^{i+1}}{||v_j^i v_l^i||} = 0, \qquad j = n+1, n+2, ..., 2n-2 \tag{7}$$

where $v_j^i v_l^i$ denotes the edge between the Steiner point v_j^i and a terminal or Steiner point v_l^i

When (7) is solved in the i-th iteration, the new locations of Steiner points are used to set up the next system of equations which is solved again in the $(i+1)$-th iteration. It can be shown that the locations of Steiner points converge to the unique, possible degenerate, RMT [15].

The numerical optimization terminates if all pairs of incident edges meet at Steiner points at angles within the interval $[2\pi/3 - \epsilon, 2\pi/3 + \epsilon]$ for an arbitrarily small constant $\epsilon > 0$. The reader is referred to [12] for the justification that a good approximation on the angles gives a good approximation of the length.

4 Heuristics

The Euclidean SMT heuristics in R^d, $d \geq 3$, suggested in this paper are modifications of the well known $O(n \log n)$ heuristic in R^2 [13, 19]. While these heuristics also work in R^2, they do not perform as well as specialized 2-dimensional heuristics. This is in particular due to the fact that there are no fast exact algorithms for finding Euclidean SMTs for small sets of terminals.

Consider a k-simplex σ, $1 \leq k \leq d$, in R^d on a subset T_σ of $k+1$ terminals of T and the subgraph of $MST(T)$ induced by T_σ. If connected, this subgraph is an MST for T_σ and is denoted MST_σ. $MST(T)$ is then said to *cover* σ.

Consider a d-sausage S_m, $m > 1$ in R^d on a subset T_{S_m} of $d+m$ terminals of T and the subgraph of $MST(T)$ induced by T_{S_m}. If connected, this subgraph is an MST for T_{S_m} and is denoted MST_{S_m}. $MST(T)$ is then said to *cover* S_m.

Let ST_σ denote a low cost Steiner tree spanning T_σ and let $\gamma_\sigma = ||ST_\sigma||/||MST_\sigma||$. Finally, let SMT_σ denote the SMT for T_σ. Define ST_{S_m}, γ_{S_m} and SMT_{S_m} in a similar way. The heuristic has the following four stages.

- For every **covered k-simplex** σ, $k = 1, 2, \ldots, d$, $1 \leq k \leq d$, in $DT(T)$, place a low cost Steiner tree $ST(T_\sigma)$ spanning T_σ on a priority queue H ordered by non-decreasing γ_σ. Ties are broken by non-decreasing $|T_\sigma|$ and, if necessary, by non-decreasing $||ST(T_\sigma)||$, see Section 5.
- For every **maximal, covered d-sausage** S_m, $d < m \leq n$, place a low cost Steiner tree $ST(T_{S_m})$ spanning T_{S_m} on the priority queue H ordered by non-decreasing γ_{S_m}. Ties are broken by non-decreasing $|T_{S_m}|$ and, if necessary, by non-decreasing $||ST(T_{S_m})||$, see Section 6.

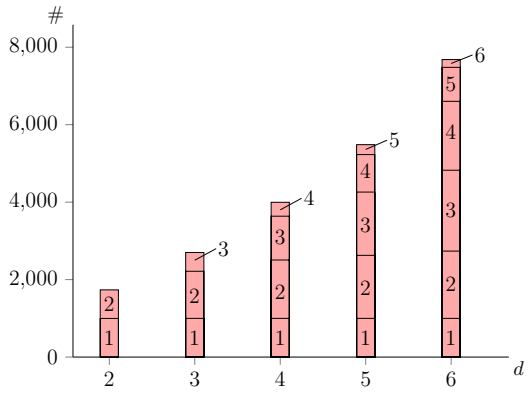

Fig. 4. Number of covered k-simplices for 1000 randomly generated points, $1 \leq k \leq d$. The bottom part shows $k = 1$, then $k = 2$, and so on.

- **Concatenation:** Create a Steiner forest $SF(T)$ (initially with no line segments). Extract Steiner trees from the priority queue H and add them one by one to $SF(T)$ unless a cycle is created, see Subsection 7.1.
- **Fine tuning:** Extend the topology of $SF(T)$ to a FST by attaching each terminal of degree b, $b \geq 2$, to one of the $b - 1$ appropriately connected Steiner points, see Subsection 7.2. Apply the numerical optimization method to obtain the RMT for this FST, see Subsection 3.2.

In order to justify the importance of covered d-simplices and d-sausages, consider the terminals T shown in Fig. 5. $DT(T)$ and $MST(T)$ are indicated on the left. In particular, 2-simplex σ with $T_\sigma = \{t_i, t_j, t_k\}$ is in $DT(T)$ but it is not covered. The Steiner ratio γ_β will be smaller than any other Steiner ratio of 2-simplices in this $DT(T)$. However, the inclusion of SMT_σ into the overall heuristic solution for T (right) would result in a very bad solution.

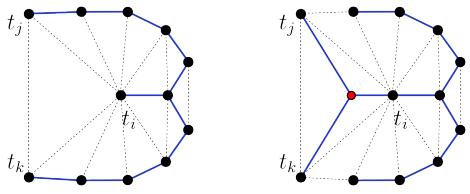

Fig. 5. MST and a poor Steiner tree resulting from using RMTs of uncovered simplices.

Regretfully, covered d-simplices and covered d-sausages become less and less frequent as d grows. Hence, relatively few Steiner trees end up in the priority queue H.

5 Low Cost Steiner Trees of Covered k-Simplices

In this section the problem of finding a low cost non-degenerate Steiner tree $ST(T_\sigma)$ for a covered k-simplex σ, $1 \leq k \leq d$, is addressed. It is of course important that the determination of ST_σ is fast as such trees have to be determined for many k-simplices. Assume that the terminals of T_σ have been relabeled to $t_1, t_2, \ldots, t_{k+1}$. This ordering can for example be inherited from the ordering of terminals in T. Let $T_j = \{t_1, t_2, \ldots, t_j\}$, $j = 1, 2, \ldots, k+1$.

If $k = 1$, then σ is a 1-simplex with 2 terminals. In this case $ST(T_\sigma) = SMT_\sigma = MST_\sigma$. Add $ST(T_\sigma)$ to the priority queue H. Note that its Steiner ratio is 1 so these edges will be at the bottom of H.

If $k = 2$, then σ is a 2-simplex with three terminals. Since the terminals of T_σ define a unique plane (unless they are collinear), $ST(T_\sigma) = SMT_\alpha$ can be found in $O(1)$ time using Melzak construction for FSTs in R^2 [10]. Add $ST(T_\sigma)$ to the priority queue H if it is non-degenerate.

In the remainder of this section it is assumed that σ is a k-simplex, $k \geq 3$, and therefore $d \geq 3$. Three different methods of obtaining low cost $ST(T_\sigma)$ are described.

5.1 Numerical Optimization (NO)

If d is not too big, say $d \leq 6$, then the approximation scheme (Section 3) will yield $ST(T_\sigma)$ such that $||ST(T_\sigma)|| \approx ||SMT_\sigma||$. For any FST $\mathcal{T}_j$ with j terminals, $3 \leq j \leq k$, of a k-simplex σ, the approximation scheme generates $2j-3$ new FSTs (see Fig. 6, left). This is repeated until FSTs $\mathcal{T}_{k+1}$ with $k+1$ terminals are generated. If k-simplices are processed bottom-up, repeated generations of FSTs of subsets of T_σ can be avoided.

The numerical optimization (Subsection 3.2) is applied to each of these $f(k+1)$ FSTs (Subsection 2.1). The best RMT found approximates SMT_σ. It is added to the priority queue H if it is non-degenerate.

5.2 Restricted Numerical Optimization (RNO)

For any FST $\mathcal{T}_j$ with the j terminals, $3 \leq j \leq k$, of a k-simplex σ, the approach described in Subsection 5.1 generates $2j-3$ new FSTs. In order to speed up this process, the RMT of $\mathcal{T}_j$ is determined. The Steiner point s in this RMT closest to t_{j+1} is identified. $\mathcal{T}_j$ is expanded only in *three* ways by splitting the edges incident with s (instead of splitting all $2j-3$ edges) as shown in Fig. 6 (right). For each of these three FSTs, their RMTs are determined. If $j = k$, the shortest of these RMTs is added to the priority queue H if it is non-degenerate. If $j < k$, the FST of the shortest RMT is then the only one that is expanded in the next

iteration. The price for reducing the number of determined RMTs is of course the quality of $ST(T_\sigma)$.

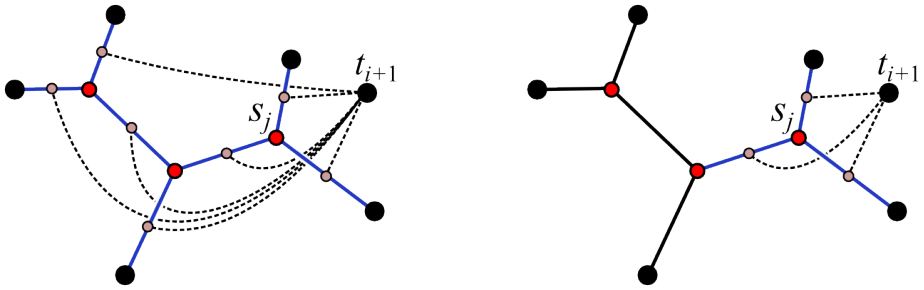

Fig. 6. Numerical optimization (left) and restricted numerical optimization (right).

5.3 Simplex Partitioning (SP)

Let σ' and σ'' be two faces of T_σ, $T_{\sigma'} \cup T_{\sigma''} = T_\sigma$, $T_{\sigma'} \cap T_{\sigma''} = \emptyset$, $|T_{\sigma'}|, |T_{\sigma''}| \geq 2$ as shown in Fig. 7. Let c'' denote the centroid of $T_{\sigma''}$. Determine a low cost Steiner tree $ST(T_{\sigma' \cup c''})$ and let s_1 denote its Steiner point adjacent to c''. Construct a low cost Steiner tree $ST(T_{\sigma'' \cup s_1})$. Let s_2 be the Steiner point adjacent to s_1. Create a Steiner tree $ST(T_\sigma)$ by replacing c'' (and its incident edge) in $ST(T_{\sigma' \cup c''})$ by $ST(T_{\sigma'' \cup s_2})$. Apply the numerical optimization (Subsection 3.2) to $ST(T_\sigma)$ for the additional length reduction. Determine another Steiner tree by swapping σ' and σ''. Repeat this for all such pairs of σ' and σ'' and add the shortest encountered Steiner tree to the priority queue H if it is non-degenerate.

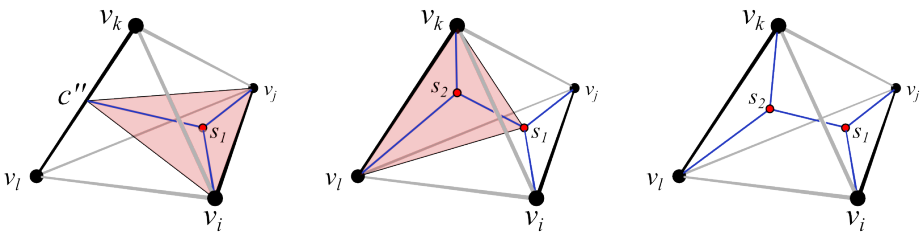

Fig. 7. Partitioning of a covered 3-simplex in R^3 using two edges.

6 Low Cost Steiner Trees for Covered *d*-Sausages

It has been observed [16], for $d = 3$ in particular, that regular d-sausages have very low Steiner ratios. In fact, their ratios are considerably less than Steiner ratios for regular d-simplices in R^d. Furthermore, the Steiner ratio decreases as the number of terminals in regular d-sausages increases. Fig. 8 shows a portion of the Euclidean SMT of an almost regular 3-sausages with terminals placed on an α-helix at positions corresponding to C_α atoms of proteins.

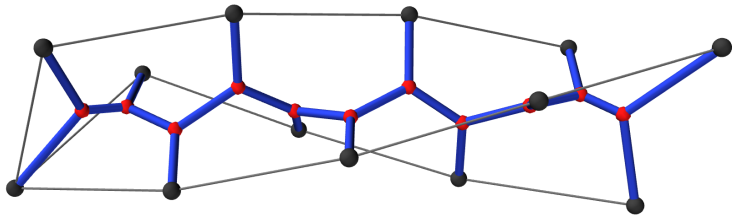

Fig. 8. Regular 3-sausage in R^3, its Euclidean SMT in blue, and MST in gray (3 MST paths meet at the end of the helix).

The notion of covered d-sausages, however, seems too weak to capture all d-sausages that could be useful when constructing low-cost Euclidean Steiner trees. This in particular becomes a problem as $d \geq 3$. On the other hand, dropping the requirement that d-sausages must be covered can result in the selection of RMTs with small ratio but with large length as already pointed out in Fig. 5.

7 Low Cost Steiner Tree for *T*

Once non-degenerate low cost Steiner trees for selected faces of d-simplices and for selected d-sausages have been stored in the priority queue H, the Steiner tree spanning all terminals of T is constructed. This is achieved by the greedy concatenation of Steiner trees stored in H and by subsequent fine tuning.

7.1 Concatenation

Non-degenerate Steiner trees stored in the priority queue H are extracted one by one and added to the Steiner forest $SF(T)$ unless they close a cycle. Initially $SF(T)$ has no line segments. The construction of $SF(T)$ is essentially the same as when constructing MST using Kruskal's algorithm [2]. Note that $SF(T)$ will be connected as H contains the edges of $MST(T)$. The Steiner ratio of these edges is 1. So, if needed, they will be added to $SF(T)$ just before H is emptied.

7.2 Fine Tuning

The tree $SF(T)$ obtained by the greedy concatenation of smaller Steiner trees may have terminals of degree higher than 1. Consider a terminal $t \in T$ and assume that it is incident with δ edges, $\delta > 1$. Let $v_1, v_2, \ldots, v_\delta$ denote the terminals and Steiner points adjacent to t. Assume that the angle $\angle v_i t v_j$ is the smallest among all angles at t between pairs of points adjacent to t. Insert a new Steiner point s adjacent to t, v_i and v_j. Remove v_i and v_j from the list of points adjacent to t and add s to it. Repeat until t is adjacent to one point. The resulting topology is an FST. Its RMT can be obtained by the numerical optimization algorithm, see Section 5.1.

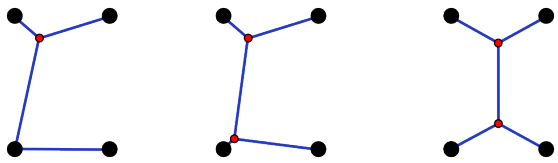

Fig. 9. Steiner points are added to all terminals with degree higher than 1 and the RMT is calculated.

8 Computational Results

Delaunay tessellation $DT(T)$ is determined in the preprocessing phase using the Qhull software package [1]. The implementation is based on the lifting algorithm [5]. Terminals in R^d are lifted to a paraboloid in R^{d+1} by adding the sum of the squares of their coordinates as the additional coordinate. The algorithm then computes the convex hull of the lifted terminals, and projects the lower convex hull back to R^d.

Once $DT(T)$ is given, $MST(T)$ is obtained using Kruskals minimum spanning tree algorithm [2]. Since the computation of $DT(T)$ and $MST(T)$ is insignificant compared to the generation of low cost Steiner trees, computational times for obtaining $DT(T)$ and $MST(T)$ are not given below.

All results presented in this paper were achieved using a Lenovo S430 laptop with an Intel Core i5-3210M CPU @ 2.50GHz, and with 4GB RAM.

8.1 Steiner Trees for Selected *k*-Simplices

Fig. 10 shows average Steiner ratios (left) and CPU-times (right) for low cost Steiner trees of 500 randomly selected d-simplices in $DT(T)$ using numerical optimization (NO) (Subsection 5.1), restricted numerical optimization (RNO) (Subsection 5.2), and simplex partitioning (SP) (Subsection 5.3) in R^d, $d = 2, 3, 4, 5, 6$. See also Table 2 for the numerical data.

For $d = 2, 3$, there is no difference between NO and RNO. SP runs much faster than both NO and RNO for $d = 2, 3$, with only a slightly worse Steiner ratio for $d = 3$. For $d = 4, 5, 6$, RNO is much faster than NO, although with somewhat worse Steiner ratios. SP achieves quite good average ratios, but it is slower than NO for $d = 4, 5$. For $d = 6$ however, SP is faster, and it appears to gain an even greater advantage over NO for larger point sets ($d > 7$), which may be beneficial when considering d-sausages (Section 6).

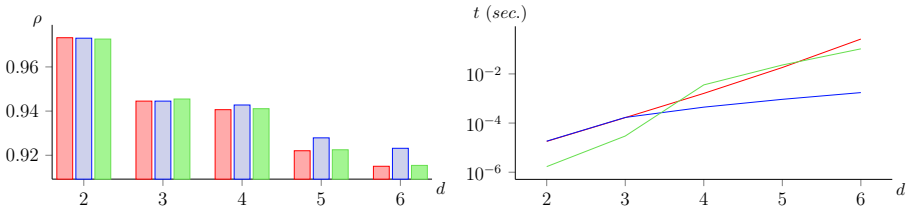

Fig. 10. Avarage Steiner ratios (left) and CPU-times (right) of NO (red), RNO (blue) and SP (green) for Steiner trees of d-simplices in R^d, $d = 2, 3, 4, 5, 6$.

8.2 Steiner Trees for Selected d-Sausages

Fig. 11 shows Steiner ratios and CPU times achieved by the heuristic with and without including selected d-sausages in the concatenation. See also Table 3 in the Appendix for the numerical data.

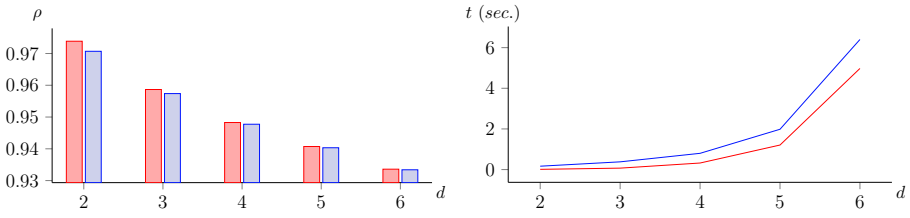

Fig. 11. Average Steiner ratios (left) and CPU-times (right) achieved for a set of 20 problem instances with 500 terminals in R^d, $d = 2, 3, 4, 5, 6$, with (blue) and without (red) including Steiner trees of covered d-sausages.

As can be seen in Fig. 12, the number of covered d-sausages decreases rapidly with the length of 3-sausages. So does the number of Steiner trees of covered 3-sausages that are included in the best solution.

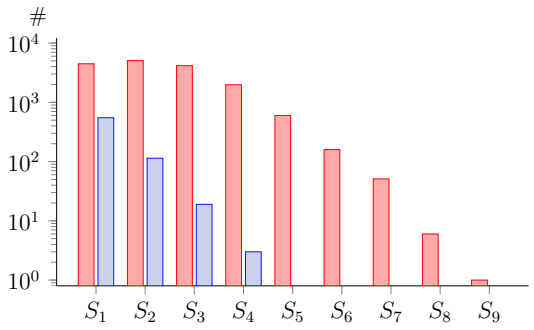

Fig. 12. Number of covered 3-sausages S_k, $k = 1 \ldots 9$, for a random instance in R^3 with $n = 10^4$ (red bars), and number of Steiner trees of 3-sausages in the concatenated solution (blue bars).

8.3 Concatenation

The greedy concatenation is very straightforward and easy to implement. However, as has been realized in the 2-dimensional case [19], the inclusion of covered d-simplices and covered d-sausages will result in solutions whose topologies are very close to the topology of the Euclidean MST. In order to address this problem, the greedy concatenation was modified in the following way. A low cost ST of an uncovered d-simplex of $DT(T)$ is placed in the front of the priority queue no matter what its length is and the concatenation is applied as previously. This is repeated to all uncovered d-simplices. The lowest of the STs is compared with the best tree found by using covered d-simplices and d-sausages. Fig. 13 shows the Steiner ratio improved (left) and how it affected CPU times. See also Table 4 in the Appendix for the numerical data.

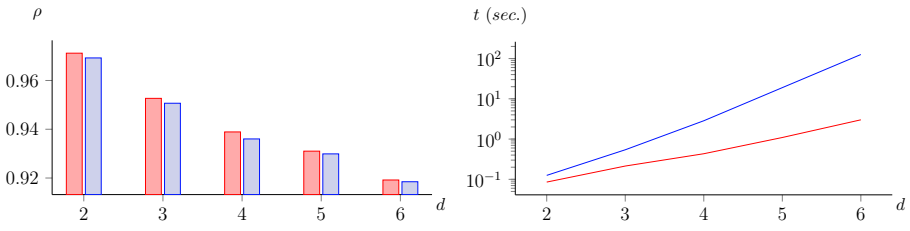

Fig. 13. Steiner ratios (left) and CPU times (right) obtained by concatenation of covered d-simplices and d-sausages (red) and by the extended concatenation (blue). The figures show the average of a computation for 20 problem instances with 200 terminals in R^d, $d = 2, 3, 4, 5, 6$.

8.4 Fine Tuning

By adding extra Steiner points and by fine tuning the tree (Section 7.2), we obtain a noticeable improvement in Steiner ratio with only minimal impact on the running time, as seen in Fig. 14. See also Table 5 for the numerical data.

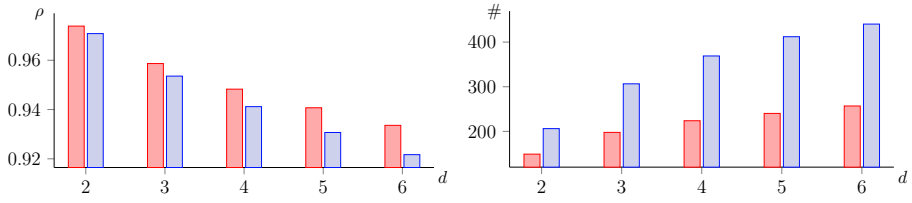

Fig. 14. Average Steiner ratios (left) obtained for a set of 20 problem instances with 500 terminals in R^d, $d = 2, 3, 4, 5, 6$, with (blue) and without (red) fine tuning, and average number of non-degenerate Steiner points (right) with (red) and without (blue) fine tuning.

8.5 Overall Performance

Fig. 15 (left) shows the Steiner ratio achieved by the heuristic for problem instances of varying dimension $d = 2, 3, 4, 5, 6$. RNO is used for determining low cost STs for the covered simplices and d-sausages (Section 6), and fine tuning is applied afterwards. Fig. 15 (right) shows a comparison of the CPU times, which increase with both dimension and set size. See Table 6 and Table 7 in the Appendix for the numerical data.

Table 1 shows the maximum size n of input instances, that the heuristic is able to handle in $\simeq 60$ seconds for $d = 2, 3, 4, 5, 6$. Again we see that the maximum input size decreases as the dimension increases.

d	2	3	4	5	6
n	150,000	80,000	41,000	15,000	3,800
ρ	0.9705	0.9534	0.9416	0.9328	0.9232

Table 1. Steiner ratios achieved for the maximum point sets in dimension $d = 2 \ldots 6$, for which the heuristic returns a result within $\simeq 60$ seconds.

Fig. 16 shows average Steiner ratios and average CPU time for the eSteiner3d benchmark instances [3] (grouped by the number of terminals). Lengths of the best solutions found by the heuristic (using RNO and fine tuning) can be seen in the Appendix in Tables 8, 9, 10, 11 and 12.

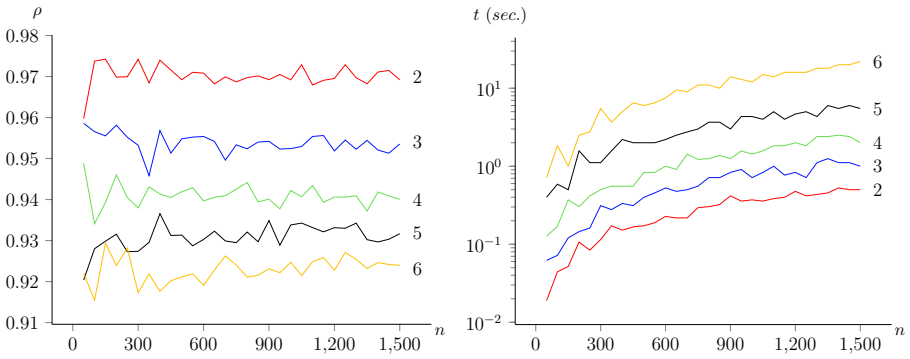

Fig. 15. Steiner ratios (left) and CPU times (right) for point sets of varying size in $d = 2 \ldots 6$.

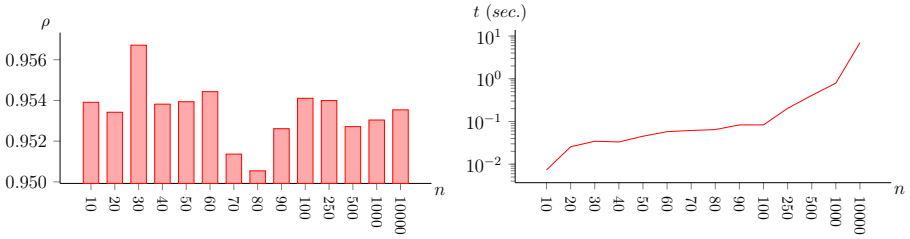

Fig. 16. Average Steiner ratios (left) and CPU times (right) for eSteiner3d benchmark instances.

Fig. 17 shows Steiner ratios and CPU times for the protein3d benchmark instances [3]. The backbone atoms of proteins form long chains. Each backbone atom (apart from the first and the last) is bonded with its predecessor, successor and a third atom or molecule (side chain). The atoms are placed such that their bonds (edges) tend to meet at 120^o. Hence, Steiner ratios for proteins should be close to 1. We also tested these instances with backbone atoms removed. Steiner points should than be places at positions corresponding to the removed atoms and the Steiner ratio should be well below 1. This indeed is the case. See Table 13 in the Appendix for the numerical data.

9 Conclusions

A family of heuristics for the d-dimensional Steiner tree problem, $d \geq 2$, has been presented. Delaunay tessellations are used to identify (covered) d-simplices and d-sausages. Low cost STs are then determined for the terminals in these d-simplices and d-sausages. Both Steiner ratio and CPU time seem to be quite promising and fairly large problems in d-dimensional spaces can be solved.

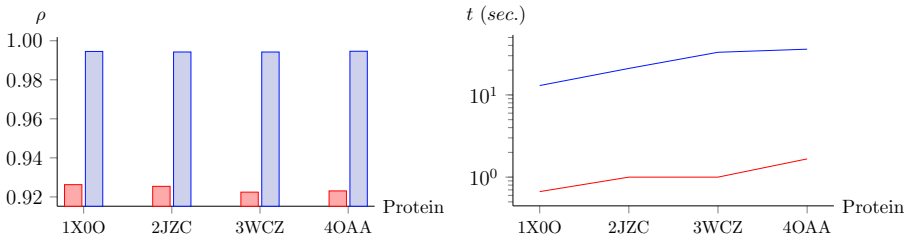

Fig. 17. Average Steiner ratios (left) and CPU times (right) for protein3d benchmark instances. Red shows the protein instances with some atoms removed, while blue shows the instances with all atoms included.

One of the interesting open problems is to extend the approach so it identifies all covered d-sausages. Our current implementation finds sequences of covered d-simplices stopping when reaching an uncovered d-simplex. However, it is possible to have a covered d-sausage with the intermediate d-simplices being uncovered.

It would also be interesting to find another way of identifying good d-sausages and d-simplices without leaning too much on the notion of being covered. Bottleneck distances [18] may prove useful in this context.

References

1. C.B. Barber, D.P. Dobkin, and H.T. Huhdanpaa, The Quickhull algorithm for convex hulls, ACM Trans. on Mathematical Software, 22 (1996) 469-483.
2. T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein, *Introduction to Algorithms*, MIT Press (2009) 3-rd ed.
3. 11 DIMACS Implementation Challenge, http://dimacs11.cs.princeton.edu/downloads.html
4. D.-Z. Du and W.D. Smith, Three disproofs of the Gilbert-Pollak conjecture on Steiner ratio in three or more dimensions, Journal of Combinatorial Theory (A) **74** (1996) 115–130.
5. H. Edelsbrunner and R. Seidel, Voronoi diagrams and arrangements, Disc. Comp. Geom. **1** (1986) 24–44.
6. M. Fampa and K.M. Anstreicher, An improved algorithm for computing Steiner minimal trees in Euclidean d-space, Discrete Optimization **5** (2008), 530-540.
7. R. Fonseca, M. Brazil, P. Winter and M. Zachariasen, Faster exact algorithms for Computing Steiner trees in higher dimensional Euclidean spaces, accepted for the 11th DIMACS Implementation Challenge on Steiner Tree Problems (2014).
8. M.R. Garey, R.L. Graham and D.S. Johnson, The complexity of computing Steiner minimal trees, SIAM J. Appl. Math. **32** (1977) 835-859.
9. E.N. Gilbert and H.O. Pollak, Steiner minimal trees, SIAM J. Appl. Math. **16** (1968) 1-29.
10. F. Hwang, D. Richards and P. Winter, *The Steiner Tree Problem*, Elsevier Science Publishers, Netherlands (1992).
11. D. Juhl, D.M. Warme, P. Winter, M. Zachariasen, The GeoSteiner software package for computing Steiner trees in the plane: An updated computational study,

accepted for the 11th DIMACS Implementation Challenge on Steiner Tree Problems (2014).
12. J.H. Rubinstein, J. Weng and N. Wormland, Approximations and lower bounds for the length of minimal Euclidean Steiner trees, J. of Global Optimization 35 (2006) 573-592.
13. J.M. Smith, D.T. Lee and J.S. Liebman, An $O(n \log n)$ heuristic for Steiner minimal tree problem on the Euclidean metric, Networks **11** (1981) 23-29.
14. J.M. Smith, R. Weiss and M. Patel, An $O(n^2)$ heuristic for Steiner minimal trees in E^3, Networks 25 (1995) 273-289.
15. W.D. Smith, How to find Steiner minimal trees in Euclidean d-space, Algorithmica **7** (1992), 137-177.
16. W.D. Smith and J.M. Smith, On the Steiner ratio in R^3, J. of Comb. Theory A 69 (1995) 301-332.
17. B. Toppur and J.M. Smith, A suasage heuristic for Steiner minimal trees in three-dimensional Euclidean space, J. of Math. Modeling and Algorithms 4 (2005) 199-217.
18. D.M. Warme, P. Winter and M. Zachariasen, Exact algorithms for plane Steiner tree problems: A computational study, in D.-Z. Du et al. (eds.) *Advances in Steiner Trees*, Kluwer Academic Press (2000) 81–116.
19. M. Zachariasen and P. Winter, Concatenation Based Greedy Heuristics for the Euclidean Steiner Tree Problem, Algorithmica (1999) 418-437.

A Tables with Numerical Result

d	2	3	4	5	6
$\overline{t_{\mathrm{NO}}}$ $(sec.)$	0.000017995	0.00016655	0.00161072	0.0182218	0.261829
$\overline{\rho_{\mathrm{NO}}}$	0.973217	0.94452	0.940643	0.922053	0.915022
$\overline{t_{\mathrm{RNO}}}$ $(sec.)$	0.000018374	0.00016942	0.00044427	0.00092436	0.0017372
$\overline{\rho_{\mathrm{RNO}}}$	0.972999	0.944516	0.942761	0.927888	0.92316
$\overline{t_{\mathrm{SP}}}$ $(sec.)$	1.69027e-06	2.96765e-05	0.00357474	0.0230367	0.104729
$\overline{\rho_{\mathrm{SP}}}$	0.972601	0.945469	0.941092	0.92249	0.915417

Table 2. Average Steiner ratios and CPU-times of NO, RNO and SP for Steiner trees of d-simplices in R^d, $d = 2, 3, 4, 5, 6$. See also Fig. 10.

d	2	3	4	5	6
$\overline{t}$ $(sec.)$	0.0117	0.0710	0.3226	1.2075	4.9750
$\overline{\rho}$	0.9739	0.9587	0.9483	0.9407	0.9336
$\overline{t^*}$ $(sec.)$	0.1663	0.3806	0.7997	1.9833	6.4000
$\overline{\rho^*}$	0.9707	0.9574	0.9477	0.9403	0.9334

Table 3. Average Steiner ratios and CPU-times achieved for a set of 20 problem instances with 500 terminals in R^d, $d = 2, 3, 4, 5, 6$, with (*) and without including Steiner trees of covered d-sausages. See also Fig. 11.

d	2	3	4	5	6
$\overline{t}$ $(sec.)$	0.0854	0.2138	0.4324	1.0940	3.0083
$\overline{\rho}$	0.9711	0.9527	0.9389	0.9310	0.9192
$\overline{t^*}$ $(sec.)$	0.1254	0.5418	2.8500	18.950	125.95
$\overline{\rho^*}$	0.9693	0.9507	0.9360	0.9299	0.9185

Table 4. Average Steiner ratios and CPU times obtained by concatenation of covered d-simplices and d-sausages and by the extended concatenation (*). The averages are computed for a set of 20 problem instances with 200 terminals in R^d, $d = 2, 3, 4, 5, 6$. See also Fig. 13.

d	2	3	4	5	6
$\overline{\rho}$	0.973864	0.958668	0.948276	0.940723	0.933587
$\overline{\#S}$	149.05	197.75	223.85	240.25	256.9
$\overline{\rho^*}$	0.97082	0.953568	0.941181	0.93069	0.921695
$\overline{\#S^*}$	206.2	306.55	368.95	412.05	440.35

Table 5. Average Steiner ratios and CPU times obtained by the heuristic with (*) and without using fine tuning after the concatenation. The averages are computed for a set of 20 problem instances with 500 terminals in R^d, $d = 2, 3, 4, 5, 6$. See also Fig. 14.

$n\backslash d$	2	3	4	5	6
50	0.959753	0.958583	0.948836	0.920395	0.921987
100	0.973755	0.956553	0.934118	0.928026	0.915472
150	0.974232	0.955526	0.93932	0.929854	0.929356
200	0.969872	0.958149	0.946037	0.931566	0.923916
250	0.969939	0.955209	0.940547	0.927293	0.928176
300	0.974192	0.953245	0.937974	0.927394	0.917285
350	0.968482	0.945841	0.9431	0.929571	0.921847
400	0.973963	0.956813	0.941372	0.936574	0.91767
450	0.971593	0.951349	0.940501	0.931255	0.9202
500	0.969242	0.954807	0.941911	0.931335	0.921126
550	0.971028	0.955235	0.942882	0.928727	0.921874
600	0.970836	0.955378	0.939663	0.930282	0.919092
650	0.968217	0.954189	0.940568	0.932296	0.922853
700	0.969926	0.949622	0.940952	0.929914	0.926253
750	0.968695	0.953336	0.942529	0.929513	0.924011
800	0.969736	0.952388	0.944168	0.932087	0.921099
850	0.970163	0.954022	0.939446	0.929722	0.921554
900	0.969251	0.954192	0.9401	0.934886	0.923131
950	0.970512	0.952332	0.937763	0.928871	0.92218
1000	0.969202	0.952428	0.942194	0.933819	0.924703
1050	0.972873	0.952946	0.940718	0.934263	0.921455
1100	0.967925	0.955391	0.943399	0.933176	0.924871
1150	0.969082	0.955615	0.939337	0.932134	0.92587
1200	0.969532	0.951863	0.940609	0.933144	0.92277
1250	0.972876	0.954515	0.940615	0.93301	0.927126
1300	0.969717	0.952274	0.940913	0.93425	0.92542
1350	0.968233	0.954433	0.937197	0.930228	0.923208
1400	0.971101	0.952076	0.941855	0.929669	0.92463
1450	0.971477	0.951313	0.940887	0.930326	0.924176
1500	0.969229	0.953547	0.940055	0.931676	0.923968

Table 6. Steiner ratios for point sets of varying size and $d = 2, 3, ..., 6$. See also Fig. 15 (left).

$n\backslash d$	2	3	4	5	6
50	0.0189036	0.0621118	0.126582	0.4	0.714286
100	0.0442478	0.0719424	0.169492	0.588235	1.83333
150	0.0518135	0.120482	0.37037	0.5	1
200	0.106383	0.144928	0.30303	1.57143	2.5
250	0.0840336	0.16129	0.416667	1.11111	2.75
300	0.114943	0.3125	0.5	1.11111	5.5
350	0.172414	0.277778	0.555556	1.57143	3.66667
400	0.151515	0.333333	0.555556	2.2	5
450	0.166667	0.3125	0.555556	2	6.5
500	0.172414	0.4	0.833333	2	6
550	0.188679	0.454545	0.833333	2	6.5
600	0.227273	0.526316	1	2.2	7.5
650	0.217391	0.47619	0.909091	2.5	9.5
700	0.217391	0.5	1.42857	2.75	9
750	0.294118	0.555556	1.22222	3	11
800	0.30303	0.714286	1.25	3.66667	11
850	0.322581	0.714286	1.375	3.66667	10
900	0.416667	0.833333	1.25	3	14
950	0.357143	0.909091	1.57143	4.33333	13
1000	0.37037	0.714286	1.42857	4.33333	12
1050	0.357143	0.833333	1.57143	4	15
1100	0.384615	1	1.83333	5	14
1150	0.4	0.769231	1.83333	4	16
1200	0.47619	0.833333	2	4.66667	16
1250	0.416667	0.714286	1.83333	5	16
1300	0.434783	1.11111	2.4	4.33333	18
1350	0.454545	1.25	2.4	6	18
1400	0.526316	1.11111	2.5	5.5	20
1450	0.5	1.11111	2.4	6	20
1500	0.5	1	2	5.5	22

Table 7. CPU times (sec) for point sets of varying size and $d = 2, 3, ..., 6$. See also Fig. 15 (right).

Name	n	t (*sec.*)	$\|\|MST\|\|$	$\|\|SMT\|\|$	ρ
estein10-00	10	0.0193237	3.33254	3.21346	0.964269
estein10-01	10	0.00152381	3.30121	3.10614	0.940908
estein10-02	10	0.00606061	3.17651	3.00851	0.947111
estein10-03	10	0.00467836	3.03209	2.85374	0.941178
estein10-04	10	0.013468	3.06878	2.95705	0.96359
estein10-05	10	0.0142857	3.41494	3.14027	0.91957
estein10-06	10	0.00508259	3.53768	3.27921	0.92694
estein10-07	10	0.0060698	3.10757	2.94078	0.946328
estein10-08	10	0.00567376	2.7329	2.62509	0.960551
estein10-09	10	0.00440529	3.1246	2.97064	0.950725
estein10-10	10	0.00715564	3.26161	3.1957	0.979792
estein10-11	10	0.00792079	3.07887	2.91954	0.948249
estein10-12	10	0.00370714	2.90027	2.82079	0.972596
estein10-13	10	0.00598802	3.21875	3.13832	0.975014
estein10-14	10	0.00459242	3.01279	2.92783	0.9718
estein20-00	20	0.0597015	4.97097	4.64244	0.933911
estein20-01	20	0.0336134	4.6131	4.49341	0.974054
estein20-02	20	0.0205128	4.94092	4.62116	0.935282
estein20-03	20	0.0130293	4.57347	4.40717	0.963637
estein20-04	20	0.0101781	4.93447	4.77452	0.967586
estein20-05	20	0.0264901	5.38111	5.24457	0.974626
estein20-06	20	0.0156863	4.81786	4.6458	0.964287
estein20-07	20	0.019802	5.20534	4.89915	0.941179
estein20-08	20	0.0210526	5.06873	4.84983	0.956812
estein20-09	20	0.0268456	4.85265	4.69029	0.966543
estein20-10	20	0.0186047	4.77163	4.55506	0.954612
estein20-11	20	0.0421053	4.60359	4.33363	0.941359
estein20-12	20	0.0325203	5.13724	4.84463	0.943041
estein20-13	20	0.031746	5.58437	5.19161	0.929668
estein20-14	20	0.0118694	4.91812	4.69534	0.954703
estein30-00	30	0.0470588	7.01432	6.67442	0.951542
estein30-01	30	0.0228571	6.65878	6.48397	0.973748
estein30-02	30	0.0481928	6.47612	6.15346	0.950176
estein30-03	30	0.0283688	6.71635	6.4346	0.95805
estein30-04	30	0.016129	6.95497	6.53684	0.93988
estein30-05	30	0.0408163	6.47014	6.35907	0.982833
estein30-06	30	0.028777	7.25329	6.86687	0.946725
estein30-07	30	0.0180995	6.15538	5.91064	0.960239
estein30-08	30	0.0264901	6.89295	6.56657	0.95265
estein30-09	30	0.0330579	6.63351	6.28676	0.947728
estein30-10	30	0.0232558	7.02758	6.80236	0.967953
estein30-11	30	0.0350877	6.57286	6.34189	0.96486
estein30-12	30	0.0140351	6.85188	6.46326	0.943284
estein30-13	30	0.0174672	6.33981	6.07511	0.958249
estein30-14	30	0.117647	7.0321	6.70007	0.952784

Table 8. Results for the eSteiner3d benchmark instances ($n = 10, 20, 30$), see Fig. 16

Name	n	t $(sec.)$	$\|\|MST\|\|$	$\|\|SMT\|\|$	ρ
estein40-00	40	0.0216216	7.66669	7.37741	0.962268
estein40-01	40	0.0272109	7.76159	7.48703	0.964626
estein40-02	40	0.0344828	8.62943	8.24817	0.955818
estein40-03	40	0.0310078	7.44513	7.12007	0.956341
estein40-04	40	0.0285714	8.04469	7.73211	0.961145
estein40-05	40	0.0181818	8.65937	8.21838	0.949074
estein40-06	40	0.0228571	8.16906	7.79646	0.954389
estein40-07	40	0.0412371	8.47427	8.1179	0.957947
estein40-08	40	0.0263158	8.96284	8.42385	0.939864
estein40-09	40	0.0139373	8.98378	8.58169	0.955243
estein40-10	40	0.034188	8.03186	7.70493	0.959296
estein40-11	40	0.0444444	7.6789	7.3721	0.960047
estein40-12	40	0.0526316	8.75273	8.27936	0.945918
estein40-13	40	0.0655738	8.1323	7.68032	0.944422
estein40-14	40	0.0347826	8.00802	7.53441	0.940858
estein50-00	50	0.028777	9.46093	9.02019	0.953415
estein50-01	50	0.0421053	9.50813	9.07295	0.95423
estein50-02	50	0.0421053	9.60233	9.066	0.944146
estein50-03	50	0.037037	9.06019	8.69446	0.959633
estein50-04	50	0.0714286	9.23986	8.85541	0.958393
estein50-05	50	0.0493827	9.48477	8.98387	0.947189
estein50-06	50	0.0357143	9.287	8.82325	0.950065
estein50-07	50	0.0357143	9.15102	8.74277	0.955387
estein50-08	50	0.032	8.6114	8.26943	0.960288
estein50-09	50	0.057971	8.96103	8.47866	0.94617
estein50-10	50	0.0634921	9.72393	9.25088	0.951352
estein50-11	50	0.0597015	8.99159	8.69897	0.967457
estein50-12	50	0.0344828	9.20068	8.74622	0.950605
estein50-13	50	0.0298507	10.3316	9.90477	0.958683
estein50-14	50	0.057971	9.87496	9.40119	0.952023
estein60-00	60	0.0625	10.1596	9.7025	0.955007
estein60-01	60	0.0283688	10.5666	9.98463	0.944924
estein60-02	60	0.0408163	10.9501	10.2708	0.937966
estein60-03	60	0.0526316	10.186	9.79076	0.961201
estein60-04	60	0.0434783	11.161	10.7229	0.960748
estein60-05	60	0.08	10.6887	10.3223	0.965725
estein60-06	60	0.0588235	11.6261	11.0394	0.949539
estein60-07	60	0.043956	11.0703	10.5441	0.952467
estein60-08	60	0.0487805	10.6514	10.1254	0.950617
estein60-09	60	0.0869565	10.6696	10.1463	0.950959
estein60-10	60	0.057971	10.3296	9.80177	0.948898
estein60-11	60	0.0769231	11.5656	11.1258	0.961977
estein60-12	60	0.040404	11.0637	10.665	0.963971
estein60-13	60	0.0930233	10.4158	9.99296	0.959402
estein60-14	60	0.0512821	11.3753	10.8412	0.953045

Table 9. Results for the eSteiner3d benchmark instances ($n = 40, 50, 60$), see Fig. 16

Name	n	t $(sec.)$	$\|\|MST\|\|$	$\|\|SMT\|\|$	ρ
estein70-00	70	0.097561	11.7109	11.0698	0.945261
estein70-01	70	0.0444444	12.0846	11.5269	0.95385
estein70-02	70	0.0555556	11.8165	11.3584	0.961232
estein70-03	70	0.0689655	11.8165	11.2597	0.95288
estein70-04	70	0.0689655	11.826	11.3189	0.957121
estein70-05	70	0.0526316	11.9274	11.3266	0.94963
estein70-06	70	0.056338	11.8306	11.2176	0.948184
estein70-07	70	0.045977	11.8438	11.222	0.947503
estein70-08	70	0.0449438	11.7712	11.1607	0.948136
estein70-09	70	0.0655738	12.1782	11.5322	0.946954
estein70-10	70	0.047619	11.9779	11.3426	0.946958
estein70-11	70	0.0754717	11.9486	11.3531	0.950162
estein70-12	70	0.0481928	11.2702	10.8418	0.961988
estein70-13	70	0.043956	12.1057	11.5076	0.950588
estein70-14	70	0.108108	12.0685	11.4652	0.950009
estein80-00	80	0.057971	13.238	12.5689	0.949461
estein80-01	80	0.0909091	12.8044	12.1268	0.947082
estein80-02	80	0.0666667	12.6707	12.0772	0.953161
estein80-03	80	0.08	11.8787	11.2431	0.946488
estein80-04	80	0.0533333	13.3848	12.7056	0.949254
estein80-05	80	0.0816327	13.7129	13.013	0.948959
estein80-06	80	0.040404	13.9391	13.2242	0.948713
estein80-07	80	0.0512821	12.8821	12.3697	0.960221
estein80-08	80	0.0689655	13.6495	12.8779	0.943477
estein80-09	80	0.0701754	13.2128	12.5116	0.946934
estein80-10	80	0.1	12.7368	12.0223	0.943903
estein80-11	80	0.0540541	12.2738	11.7939	0.960899
estein80-12	80	0.0571429	13.6879	12.9577	0.946653
estein80-13	80	0.0434783	13.4411	12.8207	0.953843
estein80-14	80	0.0526316	13.2343	12.6927	0.959078
estein90-00	90	0.043956	14.0648	13.4028	0.952934
estein90-01	90	0.0625	13.9575	13.1991	0.945663
estein90-02	90	0.0689655	14.1526	13.4659	0.951481
estein90-03	90	0.105263	13.1386	12.6454	0.962462
estein90-04	90	0.125	13.7569	13.0934	0.951766
estein90-05	90	0.0769231	13.8111	13.1283	0.950563
estein90-06	90	0.0816327	13.7461	13.0909	0.952339
estein90-07	90	0.0869565	14.3591	13.6366	0.949684
estein90-08	90	0.0909091	13.2578	12.6887	0.957069
estein90-09	90	0.0634921	14.1914	13.6027	0.958513
estein90-10	90	0.0754717	14.0389	13.3868	0.953555
estein90-11	90	0.111111	14.0244	13.2634	0.945734
estein90-12	90	0.108108	14.0708	13.4523	0.956048
estein90-13	90	0.0655738	13.4268	12.7457	0.949279
estein90-14	90	0.0816327	14.3308	13.6441	0.952081

Table 10. Results for the eSteiner3d benchmark instances ($n = 70, 80, 90$), see Fig. 16

Name	n	t (*sec.*)	$\|\|MST\|\|$	$\|\|SMT\|\|$	ρ
estein100-00	100	0.0519481	15.1935	14.5528	0.957833
estein100-01	100	0.0816327	15.0617	14.3938	0.955658
estein100-02	100	0.102564	15.4134	14.599	0.947164
estein100-03	100	0.0588235	15.8369	15.19	0.959155
estein100-04	100	0.0625	15.8219	15.023	0.949506
estein100-05	100	0.0677966	14.4905	13.7671	0.950077
estein100-06	100	0.108108	15.5273	14.6334	0.94243
estein100-07	100	0.114286	15.8046	15.1401	0.957954
estein100-08	100	0.0952381	15.4278	14.7947	0.958961
estein100-09	100	0.0714286	15.3972	14.7208	0.956066
estein100-10	100	0.0952381	15.475	14.7741	0.954708
estein100-11	100	0.0727273	15.2883	14.6879	0.960725
estein100-12	100	0.0571429	14.9266	14.2796	0.956653
estein100-13	100	0.105263	14.8634	14.1804	0.954045
estein100-14	100	0.105263	15.7385	14.9617	0.950646
estein250-00	250	0.148148	26.9975	25.7201	0.952685
estein250-01	250	0.285714	27.4092	26.1538	0.954197
estein250-02	250	0.235294	26.6891	25.4058	0.951916
estein250-03	250	0.266667	27.1691	25.9614	0.955547
estein250-04	250	0.173913	26.9568	25.7959	0.956936
estein250-05	250	0.173913	27.8833	26.6886	0.957152
estein250-06	250	0.2	27.5863	26.292	0.953081
estein250-07	250	0.142857	27.4988	26.2006	0.952789
estein250-08	250	0.222222	28.1783	26.8719	0.953635
estein250-09	250	0.190476	26.6414	25.5073	0.957429
estein250-10	250	0.235294	26.0041	24.7003	0.949864
estein250-11	250	0.222222	26.6477	25.3879	0.952726
estein250-12	250	0.25	27.0666	25.9204	0.957651
estein250-13	250	0.153846	27.3973	26.1502	0.95448
estein250-14	250	0.142857	27.4486	26.0707	0.949803
estein500-00	500	0.4	42.8324	40.8101	0.952787
estein500-01	500	0.444444	42.5191	40.5528	0.953753
estein500-02	500	0.444444	42.3945	40.3159	0.950971
estein500-03	500	0.333333	42.9536	40.9332	0.952963
estein500-04	500	0.4	42.3192	40.3893	0.954395
estein500-05	500	0.444444	42.9438	40.891	0.952196
estein500-06	500	0.4	41.9621	39.8969	0.950784
estein500-07	500	0.444444	42.0414	40.0701	0.953111
estein500-08	500	0.363636	42.7926	40.6453	0.949821
estein500-09	500	0.363636	41.7776	39.8347	0.953494
estein500-10	500	0.444444	41.3443	39.4336	0.953786
estein500-11	500	0.363636	42.9622	41.0086	0.954526
estein500-12	500	0.5	42.1053	40.2639	0.956267
estein500-13	500	0.363636	43.2336	40.9863	0.948018
estein500-14	500	0.4	42.446	40.4858	0.953819

Table 11. Results for the eSteiner3d benchmark instances ($n = 100, 250, 500$), see Fig. 16

Name	n	$t(sec.)$	$\|\|MST\|\|$	$\|\|SMT\|\|$	ρ
estein1000-00	1000	0.666667	67.0264	63.955	0.954176
estein1000-01	1000	0.8	66.6223	63.447	0.952338
estein1000-02	1000	0.8	67.2279	64.1912	0.954831
estein1000-03	1000	1	66.8637	63.6157	0.951424
estein1000-04	1000	0.666667	66.8818	63.7223	0.95276
estein1000-05	1000	0.8	66.7787	63.5926	0.952289
estein1000-06	1000	1	66.9816	63.9016	0.954016
estein1000-07	1000	0.8	67.0887	64.0233	0.954308
estein1000-08	1000	0.8	66.8243	63.7318	0.953721
estein1000-09	1000	0.666667	66.7329	63.5733	0.952653
estein1000-10	1000	0.8	68.2402	64.935	0.951565
estein1000-11	1000	0.8	66.0117	63.0112	0.954547
estein1000-12	1000	0.666667	66.2443	63.2615	0.954974
estein1000-13	1000	0.8	67.8424	64.4772	0.950398
estein1000-14	1000	0.8	67.0087	63.73	0.951071
estein10000-0	10000	7	305.303	291.118	0.953538

Table 12. Results for the eSteiner3d benchmark instances ($n = 1000, 10000$), see Fig. 16

Name	n	t $(sec.)$	$\|\|MST\|\|$	$\|\|SMT\|\|$	ρ
1X0O_all	1925	13	2391.62	2378.51	0.994516
1X0O	1121	0.667	2342.98	2170.23	0.926270
2JZC_all	3170	21	3948.22	3925.48	0.994242
2JZC	1865	1	4001.87	3703.38	0.925412
3WCZ_all	4808	33	5686.50	5653.67	0.994227
3WCZ	2829	1	5861.48	5406.89	0.922444
4OAA_all	6230	36	7381.35	7341.65	0.994623
4OAA	3594	1.667	7528.47	6949.30	0.923070

Table 13. Results for the protein3d benchmark instances, see Fig. 17