

Polynomial-Time Approximation Schemes for the Euclidean Survivable Network Design Problem ^{*}

Artur Czumaj¹, Andrzej Lingas², and Hairong Zhao¹

¹ Department of Computer Science, New Jersey Institute of Technology, Newark, NJ 07102-1982, USA, {czumaj, hairong}@cis.njit.edu

² Department of Computer Science, Lund University, Box 118, S-22100 Lund, Sweden, andrzej.Lingas@cs.lth.se

Abstract. The survivable network design problem is a classical problem in combinatorial optimization of constructing a minimum-cost subgraph satisfying predetermined connectivity requirements. In this paper we consider its geometric version in which the input is a complete Euclidean graph. We assume that each vertex v has been assigned a connectivity requirement r_v . The output subgraph is supposed to have the vertex- (or edge-, respectively) connectivity of at least $\min\{r_v, r_u\}$ for any pair of vertices v, u .

We present the first polynomial-time approximation schemes (PTAS) for basic variants of the survivable network design problem in Euclidean graphs. We first show a PTAS for the Steiner tree problem, which is the survivable network design problem with $r_v \in \{0, 1\}$ for any vertex v . Then, we extend it to include the most widely applied case where $r_v \in \{0, 1, 2\}$ for any vertex v . Our polynomial-time approximation schemes work for both vertex- and edge-connectivity requirements in time $\mathcal{O}(n \log n)$, where the constants depend on the dimension and the accuracy of approximation. Finally, we observe that our techniques yield also a PTAS for the multigraph variant of the problem where the edge-connectivity requirements satisfy $r_v \in \{0, 1, \dots, k\}$ and $k = \mathcal{O}(1)$.

1 Introduction

In this paper we consider a geometric version of the *survivable network design problem*. The survivable network design problem is a classical problem in combinatorial optimization because of its evident applications in telecommunication, communication network design, VLSI design, etc. In its most general formulation, there is given an undirected graph $G = (V, E)$, a cost function $c : E \rightarrow \mathbb{R}$, and a connectivity requirement function r mapping any pair of vertices to \mathbb{N} . The task is to find a minimum-cost subgraph of G such that for any pair of vertices $v, u \in V$, the subgraph has $r_{v,u}$ internally vertex-disjoint (or edge-disjoint, respectively) paths between v and u . Often, the output is allowed to be a multigraph [23].

In many applications of this problem, often regarded as the most interesting ones [8, 13], the connectivity requirement function is specified with the help of a one-argument function which assigns to each vertex v its connectivity type $r_v \in \mathbb{N}$. Then, for any pair

^{*} Research supported in part by NSF grant CCR-0105701, SBR grant No. 421090, and TFR grant 221-99-344.

of vertices $v, u \in V$, the connectivity requirement $r_{u,v}$ is simply given as $\min\{r_u, r_v\}$ [10–13, 20, 23]. Following the literature, we assume this standard simplification of the connectivity requirements function in this paper. Notice that, in particular, this includes the *Steiner tree problem* [21], in which $r_v \in \{0, 1\}$ for any vertex $v \in V$. It also includes the most widely applied variant of the survivability problem in which $r_v \in \{0, 1, 2\}$ for any vertex $v \in V$ (see, e.g., [13, 20, 23]).

In the geometric version of the survivable network design problem, the input vertices are points in \mathbb{R}^d and the cost of each link is equal to the Euclidean distance between its endpoints (which is a good approximation in many applications, since often the “installation” and the “service” cost is roughly proportional to the length of the link [20]). We focus on two most basic variants of the geometric survivable network design problem when the connectivity requirements satisfy $r_v \in \{0, 1\}$ or $r_v \in \{0, 1, 2\}$ for all $v \in V$. The arguments provided by Grötschel *et al.* [13] (see also [20, 23]), suggest that the second special case of the survivability problem models well many applications, e.g., the problem of designing survivable fiber telephone networks [20, 23]. In the case of fiber communication networks for telephone companies, network topologies with connectivity requirements in $\{0, 1, 2\}$ provide an adequate level of survivability for the distinguished central nodes of connectivity type 2. Simply, most failures usually can be repaired relatively quickly and, as statistical studies have revealed, it is unlikely that a second failure will occur for their duration.

1.1 New contributions

We design the *first polynomial-time approximation schemes* (PTASs) for the two aforementioned basic variants of the geometric survivable network design problem.

First, we consider the simplest case in which $r_v \in \{0, 1\}$ for any vertex $v \in V$, that is, the Steiner tree problem. We design an algorithm that, for any constant d and any constant ε , returns a Steiner tree whose cost is at most $(1 + \varepsilon)$ times larger than the minimum. The algorithm runs in time $\mathcal{O}(n \log n)$. For general d and ε , its running time is $\mathcal{O}(n \log n (d/\varepsilon)^{\mathcal{O}(d)}) + \mathcal{O}(n (d/\varepsilon)^{(d/\varepsilon)^{\mathcal{O}(d^2)}})$.

Next, we consider the case when $r_v \in \{0, 1, 2\}$ for any vertex $v \in V$; this is the classical problem investigated thoroughly by Grötschel and Monma *et al.* [10–13, 20, 23]. We extend the algorithm for the Steiner tree problem to design an algorithm that, for any constant d and any constant ε , returns a graph satisfying all the vertex (or edge, respectively) connectivity requirements and having the cost at most $(1 + \varepsilon)$ times larger than the minimum. The algorithm runs in time $\mathcal{O}(n \log n)$. When d and ε are allowed to vary arbitrarily, its running time is $\mathcal{O}(n \log n (d/\varepsilon)^{\mathcal{O}(d)}) + \mathcal{O}(n (d/\varepsilon)^{(d/\varepsilon)^{\mathcal{O}(d^2)}})$.

Finally, we observe that our techniques yield also a PTAS for the multigraph variant where the edge-connectivity requirements satisfy $r_v \in \{0, 1, \dots, k\}$ and $k = \mathcal{O}(1)$.

Our polynomial-time approximation schemes follow an approach similar to those used in the recent PTASs for finding TSP, (complete) Steiner trees, and minimum-cost biconnected spanning subgraph in Euclidean graphs, see [1, 4, 22]. However, there are many important differences that make the new results significantly more complicated. First of all, we have to deal with the restriction of the Steiner points to the set given *a priori* (unlike in the minimum-cost Euclidean (complete) Steiner trees problem, in

which Steiner points are allowed to be any points in \mathbb{R}^d). Furthermore, we have to deal with non-uniform connectivity requirements. The substantial differences and complications occur in the so called filtering phase and searching phase (dynamic programming).

As presented in the paper, all our PTASs are randomized and achieve the promised approximation guarantees and running time on the average. However, all our algorithms can be *derandomized* in a way similar to that used by Rao and Smith in [22]. The derandomization preserves the running time of $\mathcal{O}(n \log n)$ for constant d and ε . Because of space limitations we defer this discussion to the full version of this paper.

1.2 Related works

There has been a lot of research on the survivable network design problem. Typically, the research addresses either practical heuristics and algorithms (see, e.g., [2, 10], [11–13, 20, 23]) or the general problem for arbitrary networks (see, e.g., [5, 6, 8, 17, 24]), or the problem restricted to very specific networks. In particular, the celebrated result due to Jain [17] gives a polynomial-time 2-approximation algorithm for the edge-connected survivable network design problem (for *arbitrary connectivity requirements*). Also, polynomial-time 2-approximation algorithms for arbitrary networks in the case $r_{v,u} \in \{0, 1, 2\}$ for every v, u have been recently presented [5, 6]. We are not aware of any other good polynomial-time approximation algorithm specialized for the geometric version of the survivability problem except the case when $r_v \in \{0, 1\}$ for every v [21]. If $r_v \in \{0, 1\}$ for every v and $U = \{v \in V : r_v = 1\}$, then one can easily show that a minimum spanning tree of U guarantees the approximation ratio of 2 in any metric space (and thus, in particular, in any Euclidean space \mathbb{R}^d). Importantly, in this case the geometric survivability problem is a generalization of the classical *Euclidean (complete) Steiner tree problem* (see, e.g., [9, 15, 16, 21, 25]). The Euclidean (complete) Steiner tree problem for a finite set of points S in \mathbb{R}^d is to construct a minimum length tree whose vertex set consists of all points in S and possibly some other points in \mathbb{R}^d . Thus, the latter problem can be regarded as a survivable network design problem on an infinite vertex domain, i.e., $V = \mathbb{R}^d$ and $r_v = 1$ for any $v \in S$, and $r_v = 0$ otherwise. By the celebrated results due to Arora [1] and Mitchell [19] (see also [22]), the Euclidean (complete) Steiner tree problem admits a PTAS for any constant d .

Organization of the paper. Section 2 provides basic terminology used in our approximation schemes. In Section 3 we outline our new PTAS for the Steiner minimum tree problem. Section 4 outlines a generalization of the PTAS for Steiner minimum tree problem to include the survivability problem with the connectivity requirements in $\{0, 1, 2\}$. Finally, in Section 5 we briefly discuss possible further extensions of our PTASs to include the survivability problem in multigraphs with the edge-connectivity requirements in $\{0, 1, \dots, k\}$, as well as other ℓ_p^d metrics. Due to space limitations most of our technical claims and their proofs are deferred to the full version of this paper.

2 Definitions

We introduce more specific notation on Euclidean (called also geometrical) graphs. An *Euclidean graph*, is a pair $G = (P, E)$, where P is a set of points in an Euclidean space

\mathbb{R}^d and E is a subset of the pairs of points in P . Every Euclidean graph is weighted and the *cost* of edge (x, y) is equal to the Euclidean distance between points x and y . The *cost of the graph* is the sum of the costs of its edges. Additionally, we shall also consider Euclidean multigraphs, which are as Euclidean graphs but may contain parallel edges. Consistently with our definition, the edges of an Euclidean graph or multigraph $G = (P, E)$ are in one-to-one correspondence with the straight-line segments (in \mathbb{R}^d) connecting the incident vertices. (Such graphs are frequently called straight-line graphs in the literature.) Sometimes, for technical reasons, we shall also allow to *bend* some edges. A bent edge between a pair of points in P will be identified with a *straight-line path* (a path consisting of straight-line segments connecting the points).

Classes of Euclidean graphs. We shall discuss various classes of Euclidean graphs. Let P be a set of points in \mathbb{R}^d . A graph G on P is called a *t-spanner* of P , $t \geq 1$, if for any pair of points $p, q \in P$ there exists a path in G from p to q of length at most t times the Euclidean distance between p and q . Gudmundsson *et. al.* [14] gave a very efficient construction of t -spanners of small maximum degree and small cost.

Let P_0 and P_1 be sets of points in \mathbb{R}^d . An Euclidean tree is called a *Steiner tree* of P_1 if its vertex set includes all the points P_1 . All the vertices of a Steiner tree of P_1 outside P_1 are called its *Steiner points*. If the Steiner points are restricted to a point set P_0 , the tree is called a *Steiner tree of P_1 with respect to P_0* and the points in P_0 are called *Steiner point candidates*. The *Euclidean (complete) Steiner minimal tree* of P_1 is a Steiner tree of P_1 having the minimum cost. A Steiner tree of P_1 with respect to P_0 having the minimum cost will be called a *Steiner minimum tree (SMT)* of P_1 with respect to P_0 . Observe the difference between our definition of Euclidean (complete) Steiner tree and Steiner minimum tree; in this paper the abbreviation SMT is used only to denote a Steiner minimum tree.

Definition 1. (Survivable network design problem) *Let P be a set of n points in \mathbb{R}^d . Suppose that for any point $p \in P$ there is associated a value $r_p \in \{0, \dots, n-1\}$. The survivable network design problem is to find a minimum-cost Euclidean graph on P in which for any pair of points $v, u \in P$ there are at least $\min\{r_v, r_u\}$ disjoint paths between v and u . In the vertex-connected version of the problem the paths must be internally vertex-disjoint and in the edge-connected version of the problem the paths must be edge-disjoint.*

In this paper we focus mostly on the variant of the problem when $r_p \in \{0, 1, 2\}$ for any $p \in P$. We shall call this variant of the problem the $\{0, 1, 2\}$ -vertex- or -edge-connectivity problem, depending on whether the vertex-connected or the edge-connected version of the problem is considered. (Notice that the $\{0, 1, 2\}$ -vertex- and -edge-connectivity problem includes the SMT problem in which $r_p \in \{0, 1\}$ for any $p \in P$.) Furthermore, we can repeat the arguments used in [4] (which were also used earlier in [7, Section 3]) to show that in metric spaces $\{0, 1, 2\}$ -vertex-connectivity and $\{0, 1, 2\}$ -edge-connectivity are essentially equivalent (the arguments in [4] and [7] were given only for biconnectivity vs. two-edge-connectivity).

Lemma 2. *Let P_0, P_1, P_2 be any three sets of points in a metric space. Let H be a multigraph with the vertex set $P_0 \cup P_1 \cup P_2$ such that for any pair of vertices $u \in P_i$*

and $v \in P_j$ there are at least $\min\{i, j\}$, $0 \leq i, j \leq 2$, edge-disjoint paths from u to v in H . Then, in linear time, one can transform H into a graph G without increasing the total cost such that for any pair of vertices $u \in P_i$ and $v \in P_j$ there are at least $\min\{i, j\}$ internally vertex-disjoint paths from u to v in G . \square

This lemma allows us to concentrate only on the $\{0, 1, 2\}$ -edge-connectivity problem, and to allow the output to be given in a form of a multigraph.

Partitioning the space. An important component of our approximation algorithms is a partitioning scheme introduced by Arora in [1] and later extended in [3, 4].

Definition 3. (Dissection, 2^d -ary tree) Given a set S of points in \mathbb{R}^d , a bounding box of S is the smallest d -dimensional axis-parallel cube L^d containing the points in S . A (2^d -ary) dissection [1] of S is the recursive partitioning of the cube into smaller sub-cubes, called regions. Each region U^d of volume > 1 is recursively partitioned into 2^d regions $(U/2)^d$. A 2^d -ary tree (for a given 2^d -ary dissection) is a tree whose root corresponds to L^d , and whose other non-leaf nodes correspond to the regions containing at least two points from S . For a non-leaf node v of the tree corresponding to a region R , its children in the tree correspond to the 2^d regions that partition R in the dissection.

For any d -vector $\mathbf{a} = (a_1, \dots, a_d)$, where all a_i are integers $0 \leq a_i \leq L$, the \mathbf{a} -shifted dissection [1, 3] of a set X of points in the cube L^d in \mathbb{R}^d is the dissection of the set X^* in the cube $(2L)^d$ in \mathbb{R}^d obtained from X by transforming each point $\mathbf{x} \in X$ to $\mathbf{x} + \mathbf{a}$. A random shifted dissection of a set of points X in a cube L^d in \mathbb{R}^d is an \mathbf{a} -shifted dissection of X with $\mathbf{a} = (a_1, \dots, a_d)$ and the elements a_1, \dots, a_d chosen independently and uniformly at random from $\{0, 1, \dots, L\}$.

In the paper we shall also study a special class of geometric graphs with respect to a given dissection [4].

Definition 4. (r -locally-light graphs) A graph is r -locally-light [4] with respect to a shifted dissection if for each region in the dissection there are at most r edges having exactly one endpoint in the region.

The main reason of introducing this class of graphs is that while it is \mathcal{NP} -hard to solve the $\{0, 1, 2\}$ -vertex- or -edge-connectivity problem (or even the minimum-cost Steiner tree problem) for arbitrary Euclidean graphs, we can show how to solve this problem efficiently for r -locally-light graphs in Section 4.

3 Steiner Minimum Tree Problem

In our attempt to provide an efficient approximation scheme for the $\{0, 1, 2\}$ -connectivity problem in Euclidean graphs, we consider the SMT problem first. We apply a method that can be seen as a combination of the approach of Arora [1] and Rao and Smith [22] developed to design a PTAS for the TSP problem with the approach of Czumaj and Lingas [4] developed to design a PTAS for k -connectivity problems. Our algorithms is based on efficient implementations of the following three procedures.

Filtering: Let P_0 and P_1 be sets of points in \mathbb{R}^d and let t be any positive real number.

Find a subset X of P_0 that satisfies the following two properties:

- The cost of the SMT of P_1 with respect to X is at most $1 + t$ times the cost of the SMT of P_1 with respect to P_0 .
- The cost of the minimum spanning tree of $X \cup P_1$ is upper bounded by $\lambda_{d,t}$ times the cost of the minimum spanning tree of P_1 , where $\lambda_{d,t}$ is a function of d and t only ($\lambda_{d,t}$ will be set to $2^{\mathcal{O}(d^4)}/t^{\mathcal{O}(d)}$).

Lightening: Let P_0 and P_1 be sets of points in \mathbb{R}^d and let t be any positive number.

Let G be any $(1+t)$ -spanner of $P_0 \cup P_1$ satisfying the so called $(t', 1+t)$ -leapfrog property [14] for $1 < t' < 1+t$. Modify G to obtain an r -locally-light graph with the vertex set $P_0 \cup P_1$ that has as its subgraph a Steiner tree of P_1 with respect to P_0 whose cost is at most $(1+2t)$ times the cost of the SMT of P_1 with respect to P_0 .

Searching: Let P_0 and P_1 be sets of points in \mathbb{R}^d and let r be any positive integer. Let G be any r -locally-light graph on $P_0 \cup P_1$. Find a minimum-cost Steiner tree of P_1 with respect to P_0 that is a subgraph of G .

3.1 Filtering for SMT

In this section we show how to perform the filtering phase efficiently. We first prove that the following algorithm finds a subset X of P_0 that satisfies the required filtering property with $t = \frac{3}{2}\epsilon$.

1. Build a $(1+\epsilon)$ -spanner S on P_1 with $n \cdot \xi_{d,\epsilon}$ edges whose total cost is upper bounded by $\xi_{d,\epsilon}$ times the cost of the minimum spanning tree of P_1 , where $\xi_{d,\epsilon} = (d/\epsilon)^{\mathcal{O}(d)}$, [14].
2. For each edge e of the spanner whose cost exceeds the $|P_1|^{-4}$ fraction of the cost of minimum spanning tree of P_1 , circumscribe a d -dimensional ball $B\langle e \rangle$ with the center at the middle of e and of radius $R\langle e \rangle$ equal to ρ_d/ϵ times the length of the edge, where ρ_d is a function depending only on d , $\rho_d = 2^{\mathcal{O}(d^3)}$.
3. Let Y be a subset of P_0 that includes all points contained in the constructed balls and possibly some other points in P_0 at distance at most $4R\langle e \rangle$ from the center of such a ball $B\langle e \rangle$.
4. For each ball $B\langle e \rangle$ define a (rectilinear) d -dimensional cube $C\langle e \rangle$ of side length $8R\langle e \rangle$ that is co-centric with the ball $B\langle e \rangle$. Within each cube $C\langle e \rangle$ introduce a grid with interspacing $|e|\epsilon^2/(8\Delta\rho_d\sqrt{d})$, where Δ is the bound of maximum degree of any MST of n points in dimension d . Let set X be initially empty. Repeatedly, in the increasing length order of the edges e of S , assign each point $p \in Y$ associated with ball $B\langle e \rangle$ to the closest point of the grid $C\langle e \rangle$. For each grid point in $C\langle e \rangle$ if there is at least one point $p \in Y$ assigned to it, add one such a point to X .

Lemma 5. (SMT Filtering) For any point sets P_0 and P_1 in \mathbb{R}^d and any positive real number ϵ , the subset X of P_0 satisfies the filtering property. \square

The proof of the lemma above is highly non-trivial and very involved. It is based on a very subtle analysis of properties of spanners and optimal Steiner trees and some

ideas from [22]. Because of space limitations we defer the proof to the full version of the paper.

The construction of the set X described above has been specially tuned to enable an efficient implementation.

Lemma 6. *The algorithm above can be implemented to determine the set X in time $(d/\epsilon)^{\mathcal{O}(d)} \cdot n \log n$, where $n = |P_0 \cup P_1|$.*

Proof. By [14], Step 1 can be implemented in time $(d/\epsilon)^{\mathcal{O}(d)} \cdot n + \mathcal{O}(d \cdot n \cdot \log n)$.

To implement Steps 2, 3, i.e., to determine the set Y , we use a core data structure for *approximate point location in equal balls* due to Indyk and Motwani [18]. For a given approximation factor $c \geq 1$ and a given real r , they designed a static data structure for a set of points $S \subseteq \mathbb{R}^d$, called (c, r) -PLEB, such that for any point $q \in \mathbb{R}^d$, if S contains a point within distance r from q then (c, r) -PLEB outputs a point $p \in S$ that is promised to be within a distance at most cr from q . Indyk and Motwani [18, Theorem 3] show that there is an algorithm for $(2, r)$ -PLEB that after an $\mathcal{O}(|S| 2^{\mathcal{O}(d)})$ -time preprocessing achieves $\mathcal{O}(d)$ query time. Note that in our algorithm above, the costs of the spanner edges from which the balls originate fall into a logarithmic number of intervals of the form $[2^i l_0, 2^{i+1} l_0)$, where l_0 is their minimum cost (which is lower bounded by the cost of the minimum spanning tree of P_1 over $|P_1|^4$). To determine Y , for $i = 0, \dots, \mathcal{O}(\log n)$, we build the $(2, 2^{i+1} l_0 \rho_d/\epsilon)$ -PLEB data structure for all the center points of spanner edges having cost in the interval $[2^i l_0, 2^{i+1} l_0)$, and then query it with all the points in P_0 . The time needed for the construction of the logarithmic number of PLEB data structures is $\mathcal{O}(\log n)$ times the number of spanner edges (which is $n \xi_{d,\epsilon} = n (d/\epsilon)^{\mathcal{O}(d)}$) and the time needed for the $|P_0|$ queries is $\mathcal{O}(\log n) \times |P_0| \times \mathcal{O}(d)$ [18]. Hence, the total time required in Steps 2 and 3 is $\mathcal{O}(n (d/\epsilon)^{\mathcal{O}(d)} \log n)$.

Observe that during the construction of Y , we can also insert each point accounted to Y into a list corresponding to the smallest ball it belongs to (approximately) by processing the $\mathcal{O}(\log n)$ PLEB queries without increasing the asymptotic time performance. These lists are useful in the implementation of Step 4. Simply, for each of the balls $B\langle e \rangle$, and for each point p in the corresponding list $L\langle e \rangle$, we check whether or not the point p after rounding off to the nearest point on the grid $C\langle e \rangle$ is already marked. If not, we mark the grid point and add p to X . It is easy to see that in this way Step 4 can be implemented in time $\mathcal{O}((d/\epsilon)^{\mathcal{O}(d)} \cdot n)$. \square

3.2 Lightning for SMT

For the Lightning phase we use a framework developed in [4] to transform spanners into r -locally-light graphs maintaining connectivity properties. Using this framework we can prove the following lemma.

Lemma 7. *Let P_0 and P_1 be sets of points in \mathbb{R}^d and let $\epsilon > 0$. Let $r = (d/\epsilon)^{\mathcal{O}(d^2)}$. Let G be any $(1 + \frac{1}{4}\epsilon)$ -spanner of $P_0 \cup P_1$ that has $n (d/\epsilon)^{\mathcal{O}(d)}$ edges, whose total cost is upper bounded by $(d/\epsilon)^{\mathcal{O}(d)}$ times the cost of the minimum spanning tree of P_1 and that satisfies the $(t, 1 + \frac{1}{4}\epsilon)$ -leapfrog property, where $1 < t < 1 + \frac{1}{4}\epsilon$. Then, one can transform G to obtain a graph H with vertex set $P_0 \cup P_1$ (i) that is r -locally-light*

and (ii) that contains as its subgraph a Steiner tree of P_1 with respect to P_0 whose cost is at most $(1 + \frac{1}{2} \epsilon)$ times the cost of the SMT of P_1 with respect to P_0 . Moreover, this transformation can be performed in time $\mathcal{O}(d^{3/2} n \log n) + n(2^{d^{\mathcal{O}(d)}} + (d/\epsilon)^{\mathcal{O}(d)})$. \square

Informally, the $(t, 1 + \frac{1}{4} \epsilon)$ -leapfrog property means that if there is an edge between u and v in the $(1 + \frac{1}{4} \epsilon)$ -spanner then any path in the spanner between u and v that does not use the edge (u, v) has cost greater than t times the cost of (u, v) . For a formal definition see Gudmundsson *et. al.* [14] where also an efficient construction of the spanner used in Lemma 7 is given.

Remark 8. The key property of the construction in Lemma 7 is that the obtained graph H has $P_0 \cup P_1$ as its vertex set. This distinguishes it from previous constructions, e.g., [1, 22], where a related graph H was allowed to use arbitrary points outside $P_0 \cup P_1$. This property is critical in the approximation of SMT and other connectivity problems (in contrast to, e.g., TSP approximation). Indeed, suppose that H has as a vertex a point $q \notin P_0 \cup P_1$ which is a cut-vertex in H and that H contains some tree T to be pruned to a Steiner tree for P (or its subset). Then, if the degree of q in T is very high, it might be impossible to remove q from T to obtain a tree of the cost as good (or almost as good) as the cost of T (in contrast, TSP on a superset of P can be easily modified to obtain a TSP on P without any cost increase). Observe that this construction allows to bend the edges, see our discussion at the beginning of Section 2. \spadesuit

3.3 Searching for SMT

Our approach in the Searching phase is to apply dynamic programming to find an optimal Steiner tree in a r -locally-light graph. Using dynamic programming approach essentially the same as the one used in PTAS algorithms for TSP (and the Euclidean complete Steiner tree problems) due to Arora [1, 22], we can prove the following result.

Lemma 9. *Let P_0 and P_1 be sets of jointly n points in \mathbb{R}^d and let r be any integer. Let G be an r -locally-light (with respect to a certain given shifted dissection) graph on $P_0 \cup P_1$. Then, in time $\mathcal{O}(n \cdot r^{\mathcal{O}(2^d r)})$ one can find a minimum-cost Steiner tree of P_1 with respect to P_0 that is a subgraph of G .* \square

3.4 Polynomial-Time Approximation Scheme for SMT

Now, we show how to combine all our arguments from Sections 3.1–3.3 to obtain a PTAS for the Euclidean SMT problem. The input to our problem consists of two sets P_0 and P_1 of points in \mathbb{R}^d of total size $|P_0| + |P_1| = n$. Our goal is to find a Steiner tree of P_1 with respect to P_0 whose cost is less than or equal to $1 + \epsilon$ times the minimum.

We first apply Lemma 5 to find a subset X of P_0 having the promised properties (with $t = \frac{1}{4} \epsilon$). Then, we take a $(1 + \frac{1}{4} \epsilon)$ -spanner G for $X \cup P_1$ and apply Lemma 7 to modify G in order to obtain an r -locally-light graph H that has as a subgraph a Steiner tree of P_1 with respect to X whose cost is upper bounded by $(1 + \frac{1}{2} \epsilon)$ times the cost of the SMT of P_1 with respect to X . Finally, we apply Lemma 9 to find a minimum-cost Steiner tree of P_1 with respect to X that is a subgraph of H and output it. This leads to the following theorem.

Theorem 10. *There exists a polynomial-time approximation scheme for the Steiner Minimum Problem in Euclidean space \mathbb{R}^d . In particular, for any sets of points P_0 and P_1 in \mathbb{R}^d of total size n , in time $\mathcal{O}(n \log n (d/\varepsilon)^{\mathcal{O}(d)}) + \mathcal{O}(n (d/\varepsilon)^{(d/\varepsilon)^{\mathcal{O}(d^2)}})$ one can find a Steiner tree of P_1 with respect to P_0 whose cost is at most $1+\varepsilon$ times the minimum. For constant d and ε , the running time of this algorithm is $\mathcal{O}(n \log n)$. \square*

4 $\{0, 1, 2\}$ -Connectivity Problem

One can extend the algorithm from the previous section to obtain a polynomial-time approximation scheme for the $\{0, 1, 2\}$ -Connectivity Problem in Euclidean graphs. Actually, we have paid special attention to present the algorithm for the SMT problem in a form extendable to include the $\{0, 1, 2\}$ -Connectivity Problem.

Due to Lemma 2, we may consider only the $\{0, 1, 2\}$ -edge-connectivity problem and allow the output to be given in a form of a multigraph. Our algorithm uses similar three phases as the algorithm for the SMT problem. The Filtering phase is essentially the same as for the SMT problem except that we work on the spanner of $P_1 \cup P_2$ in order to find X and Y . For the Lighting phase, we argue analogously as for the SMT¹.

Lemma 11. *Let P_0, P_1 and P_2 be sets of points in \mathbb{R}^d . Let $\epsilon > 0$ and $r = (d/\epsilon)^{\mathcal{O}(d^2)}$. Let G be any $(1 + \frac{1}{4}\epsilon)$ -spanner of $P_0 \cup P_1 \cup P_2$ that has $n (d/\epsilon)^{\mathcal{O}(d)}$ edges, whose total cost is upper bounded by $(d/\epsilon)^{\mathcal{O}(d)}$ times the cost of the minimum spanning tree of $P_1 \cup P_2$, and that satisfies the $(t, 1 + \frac{1}{4}\epsilon)$ -leapfrog property, where $1 < t < 1 + \frac{1}{4}\epsilon$. Then, in time $\mathcal{O}(d^{3/2} n \log n) + n(2^{d^{\mathcal{O}(d)}} + (d/\epsilon)^{\mathcal{O}(d)})$, one can transform G to obtain a graph H with vertex set $P_0 \cup P_1 \cup P_2$ that is r -locally-light and for which there is a sub-multigraph M (i) whose induced graph is H , (ii) that satisfies the connectivity requirement of every vertex in M , and (iii) whose cost is at most $(1 + \frac{1}{2}\epsilon)$ times the cost of the minimum-cost multigraph having the same connectivity property. \square*

Remark 12. One can modify this lemma to hold for arbitrary $\{0, 1, \dots, k\}$ -edge-connectivity in multigraphs with the same time complexity, provided that k is a constant. \spadesuit

Searching phase is the only phase that is completely different and rather tricky.

4.1 Dynamic Programming for $\{0, 1, 2\}$ -edge-connectivity

Consider a point set $P = P_0 \cup P_1 \cup P_2$ in \mathbb{R}^d , where P_i is the set of points whose connectivity requirement is i . For an arbitrary shifted dissection, let G be an Euclidean graph on P that is r -locally-light with respect to this dissection. We apply dynamic programming approach to find in time $\mathcal{O}(n \cdot r^{\mathcal{O}(2^d r)})$ the minimum-cost $\{0, 1, 2\}$ -edge-connected multigraph H on P whose induced graph¹ is a subgraph of G .

The main idea of our approach is similar to the method used by Arora in [1] and in the follow-up papers (see, e.g., [3, 4, 22]). We apply dynamic programming to find an optimal solution by finding optimal solutions to subproblems induced by all regions in the shifted dissection, in a bottom-up manner. That is, for each region in the dissection

¹ The graph induced by a multigraph M is the graph obtained by reducing the multiplicity of each edge of M to one.

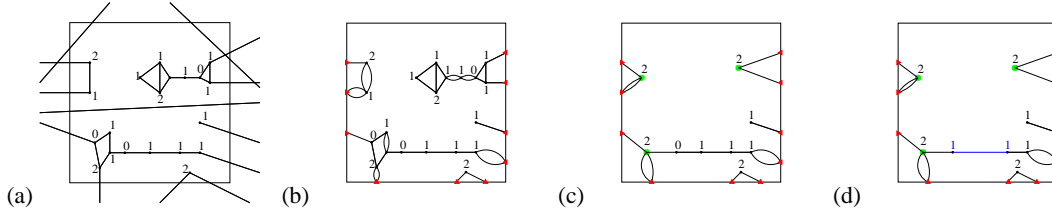


Fig. 1. (a) A part of graph G inside a region R , (b) a multigraph M_R within R , (c) contraction of two-edge-connected components in M_R , and (d) contraction of paths in M_R .

we find in particular all solutions in a form of a “sub”-multigraph on the edges (possibly duplicated) of the input graph within the region that may be extended to a minimum-cost $\{0, 1, 2\}$ -edge-connected multigraph on P . For a non-leaf region, we do it by combing solutions for the 2^d child subregions in the dissection. To reduce the number of solutions considered, we introduce various “connectivity types” for each region, and produce solely a single optimal multigraph for each of the types.

For any region R of the dissection, after duplicating some of the edges of G , a *multigraph within R* is the part of the multigraph contained within R resulting from the removal of all edges that cross R and have no endpoint inside R (see, Figure 1 (a–b)). The multigraph within R has two types of points, those from P , which we call the *input points*, and those defined by the crossings of the edges with the border of P , which we shall call the *border points*.

The aforementioned *connectivity type* of a multigraph within any region R is obtained by (see also, Figure 1 (c–d)):

- Contracting each maximal two-edge-connected component in the multigraph induced by its input points into a single vertex, and associating with the new vertex the connectivity requirement equal to the maximum connectivity requirement among all input points of the component.
- Contracting each maximal path composed of input points of degree two into the single edge with the endpoints being the first and the last vertex at the path, and associating with each endpoint of the new edge the connectivity requirement equal to the maximum connectivity requirement among all the points on the path.

Two multigraphs H_1 and H_2 within a region R are said to have the same connectivity type if their connectivity types admit isomorphism preserving the border points.

To apply dynamic programming we need the following lemma which ensures that among all multigraphs within any region in the dissection it is sufficient to consider only those having minimum cost with respect to some connectivity type.

Lemma 13. *If H is the minimum-cost $\{0, 1, 2\}$ -edge-connected multigraph on P whose induced graph is a subgraph of G , then for any region R in the shifted dissection the sub-multigraph H_R of H within R has the minimum-cost over all multigraphs within R that have the same connectivity type as H_R . \square*

If one ignores parallel edges having one endpoint as a border point and another as an input point, any connectivity type is a forest with no path having three consecutive

vertices of degree 2. Furthermore, since G is r -locally light, any region R in the dissection is crossed by at most r edges with one endpoint in R . Therefore, the number of connectivity types within R is upper bounded by $r^{\mathcal{O}(r)}$.

Our dynamic programming procedure determines for each region and for each possible connectivity type, the minimum-cost multigraph of this type within the region in a bottom-up fashion, in a similar way as it is done in [1, 3, 4, 22] and in Section 3.3. We can compute the optimal solution for each connectivity type in a region containing only a single point from P in $r^{\mathcal{O}(r)}$ time. Then, for each region containing more than a single point we compute the optimal solution for each connectivity type in the region by considering all sub-regions in total time $\mathcal{O}(2^d r) \cdot (r^{\mathcal{O}(r)})^{2^d} = r^{\mathcal{O}(2^d r)}$. Summarizing, the total time required by the dynamic programming is $\mathcal{O}(n \cdot r^{\mathcal{O}(2^d r)})$.

Hence, we obtain the following theorem.

Theorem 14. *There exists a polynomial-time approximation scheme for the $\{0, 1, 2\}$ -vertex/edge-connectivity problem in Euclidean space \mathbb{R}^d . For $\varepsilon > 0$ and any sets of points P_0, P_1, P_2 in \mathbb{R}^d of total size n , the algorithm in time $n \log n (d/\varepsilon)^{\mathcal{O}(d)} + n (d/\varepsilon)^{(d/\varepsilon)^{\mathcal{O}(d^2)}}$ finds a graph on $P_1 \cup P_2$ with possible Steiner points in P_0 that satisfies the connectivity requirement for any pair of points and whose cost is at most $1 + \varepsilon$ times the minimum.*

For constant d and ε , the running time of this algorithm is $\mathcal{O}(n \log n)$. □

5 Extensions

1. All the arguments from Section 4 except the dynamic programming part can be easily generalized to include $\{0, 1, \dots, k\}$ -edge-connectivity for multigraphs. Regarding the dynamic programming, we can combine the dynamic programming for the k -edge-connectivity from [3] with our method of dealing with non-uniform connectivity requirements. Thus, the variant of the geometric survivability problem, where the edge-connectivity requirements satisfy $r_v \in \{0, 1, \dots, k\}$ and $k = \mathcal{O}(1)$, admits also a PTAS. The running time is however, significantly greater than $\mathcal{O}(n \log n)$, though it is still polynomial for constant d and ε .
2. Using the same arguments as in [1], we can extend our PTASs to include other ℓ_p^d metrics as well.
3. Our PTASs could be also extended to an infinite domain for Steiner candidate points (e.g., if $P_0 = \mathbb{R}^d$ we have the Euclidean complete Steiner problem) provided we can determine the set Y (or its good approximation) as fast as claimed in Lemma 6.

References

1. S. Arora. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *J. Assoc. Comput. Mach.*, 45(5):753–782, 1998.
2. S. Chopra and C.-Y. Tsai. A branch-and-cut approach for minimum cost multi-level network design. *Discrete Math.*, 242:65–92, 2002.
3. A. Czumaj and A. Lingas. On approximability of the minimum-cost k -connected spanning subgraph problem. In *Proc. 10th ACM-SIAM SODA*, pp. 281–290, 1999.

4. A. Czumaj and A. Lingas. Fast approximation schemes for Euclidean multi-connectivity problems. In *Proc. 27th ICALP*, pp. 856–868, 2000.
5. L. Fleischer. A 2-approximation for minimum cost $\{0, 1, 2\}$ vertex connectivity. In *Proc. 8th IPCO*, pp. 115–129, 2001.
6. L. Fleischer, K. Jain, and D. P. Williamson. An iterative rounding 2-approximation algorithm for the element connectivity problem. In *Proc. 42nd FOCS*, pp. 339–347, 2001.
7. G. N. Frederickson and J. Jájá. On the relationship between the biconnectivity augmentation and Traveling Salesman Problem. *Theoret. Comput. Sci.*, 19(2):189–201, 1982.
8. H. N. Gabow, M. X. Goemans, and D. P. Williamson. An efficient approximation algorithm for the survivable network design problem. *Math. ProgrammingB*, 82:13–40, 1998.
9. E. N. Gilbert and H. O. Pollak. Steiner minimal trees. *SIAM J. Appl. Math.*, 16(1):1–29, 1968.
10. M. Grötschel and C. L. Monma. Integer polyhedra arising from certain network design problems with connectivity constraints. *SIAM J. Discr. Math.*, 3(4):502–523, 1990.
11. M. Grötschel, C. L. Monma, and M. Stoer. Computational results with a cutting plane algorithm for designing communication networks with low-connectivity constraints. *Operations Research*, 40(2):309–330, 1992.
12. M. Grötschel, C. L. Monma, and M. Stoer. Polyhedral and computational investigations for designing communication networks with high survivability requirements. *Operations Research*, 43:1012–1024, 1995.
13. M. Grötschel, C. L. Monma, and M. Stoer. Design of survivable networks. In M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser, editors, *Handbooks in Operations Research and Management Science*, volume 7: Network Models, chapter 10, pp. 617–672. North-Holland, Amsterdam, 1995.
14. J. Gudmundsson, C. Levcopoulos, and G. Narasimhan. Improved greedy algorithms for constructing sparse geometric spanners. In *Proc. 7th SWAT*, pp. 314–327, 2000.
15. F. K. Hwang and D. S. Richards. Steiner tree problems. *Networks*, 22:55–89, 1991.
16. F. K. Hwang, D. S. Richards, and P. Winter. *The Steiner Tree Problem*. North-Holland, Amsterdam, 1992.
17. K. Jain. A factor 2 approximation algorithm for the generalized Steiner network problem. *Combinatorica*, 21(1):39–60, 2001.
18. P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proc. 30th ACM STOC*, pp. 604–613, 1998.
19. J. S. B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, k -MST, and related problems. *SIAM J. Comput.*, 28(4):1298–1309, August 1999.
20. C. L. Monma and D. F. Shallcross. Methods for designing communications networks with certain two-connected survivability constraints. *Operations Research*, 37(4):531–541, July 1989.
21. H. J. Prömel and A. Steger. *The Steiner Tree Problem. A Tour Through Graphs, Algorithms and Complexity*. Vieweg Verlag, Wiesbaden, 2002.
22. S. B. Rao and W. D. Smith. Approximating geometrical graphs via “spanners” and “banyans.” In *Proc. 30th ACM STOC*, pp. 540–550, 1998.
23. M. Stoer. *Design of Survivable Networks*, volume 1531 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 1992.
24. D. P. Willimason, M. X. Goemans, M. Mihail, and V. V. Vazirani. A primal-dual approximation algorithm for generalized Steiner network problem. *Combinatorica*, 15:435–454, 1995.
25. P. Winter. Steiner problem in networks: A survey. *Networks*, 17:129–167, 1987.