University Library



A gateway to Melbourne's research publications

Minerva Access is the Institutional Repository of The University of Melbourne

Author/s:
Brazil, M;Volz, M;Zachariasen, M;Ras, C;Thomas, D

Title:
Computing minimum 2-edge-connected Steiner networks in the Euclidean plane

Date:
2019-01-01

Citation:
Brazil, M., Volz, M., Zachariasen, M., Ras, C. & Thomas, D. (2019). Computing minimum 2-edge-connected Steiner networks in the Euclidean plane. Networks, 73 (1), pp.89-103. https://doi.org/10.1002/net.21835.

Persistent Link:
https://hdl.handle.net/11343/284246

Check for updates

Brazil Marcus (Orcid ID: 0000-0001-8985-3752)

Computing minimum 2-edge-connected Steiner networks in the Euclidean plane

Marcus Brazil; Marcus Volz; Martin Zachariasen; Charl Ras; Doreen Thomas

Abstract

We present a new exact algorithm for computing minimum 2-edge-connected Steiner networks in the Euclidean plane. The algorithm is based on the GeoSteiner framework for computing minimum Steiner trees in the plane. Several new geometric and topological properties of minimum 2-edge-connected Steiner networks are developed and incorporated into the new algorithm. Comprehensive experimental results are presented to document the performance of the algorithm which can reliably compute exact solutions to randomly generated instances with up to 50 terminals - doubling the range of existing exact algorithms. Finally, we discuss the appearance of Hamiltonian cycles as solutions to the minimum 2-edge-connected Steiner network problem.

Keywords: 2-connected Steiner networks, 2-edge-connected networks, Geosteiner, exact algorithm, computational geometry

1 Introduction

The Euclidean 2-edge-connected Steiner network problem asks us to find a network N of minimum possible length that is 2-edge-connected, and spans a given finite set X of points (known as *terminals*) with specified locations in the Euclidean plane. The term "2-edge-connected" means that there are at least two edge-disjoint paths in N between every pair of terminals. This is also equivalent to saying that the network remains connected if any edge of N is deleted. Since a 2-edge-connected minimum-length network is necessarily 2-vertex-connected when the distance function is a metric (14), we use the shorthand "2-connected" throughout the remainder of this paper.

A solution to an instance of the 2-connected Steiner network problem is referred to as a *minimum 2-connected Steiner network* (M2SN). An important property of M2SNs is that they can contain vertices other than the given terminals; these additional vertices are referred to as *Steiner points*. The task of optimally locating these Steiner points in the plane makes the 2-connected Steiner network problem, at least in part, a geometric problem.

The 2-connected Steiner network problem was shown to be NP-hard by Luebke and Provan in (13). Despite this, however, there are useful structural properties exhibited by M2SNs, that suggest that it may be possible to design algorithms which are able to exactly solve large instances of the problem in a reasonable time. In particular, M2SNs can be decomposed into so-called *full components*. Each full component is a *full minimum Steiner tree* (FST) on a

This is the author manuscript accepted for publication and has undergone full peer review but has not been through the copyediting, typesetting, pagination and proofreading process, which may lead to differences between this version and the Version of Record. Please cite this article as doi: 10.1002/net.21835

This article is protected by copyright. All rights reserved.

Author Manuscript

Author Manuscript

subset of the terminals, where a *minimum Steiner tree* is defined to be a network of minimum total length interconnecting a given set of terminals (and possibly containing Steiner points) and where a minimum Steiner tree is said to be *full* if each terminal in the tree has degree one. An M2SN is constructed from its full components by taking a union of FSTs that are disjoint except possibly at terminals.

This decomposition property of M2SNs, which is also shared by minimum Steiner trees (without 2-connectedness), is the basis for the strategy behind *GeoSteiner*, the best known exact algorithm for constructing either minimum Steiner trees or M2SNs (for the geometric versions of these problems) for any given set of terminals (see (18) and Section 1.4 of (2)). Our main aim in this paper is to establish a number of new structural geometric properties of M2SNs which can be used to demonstrably and significantly improve the best previous implementation of GeoSteiner for M2SNs, in (19).

The main contributions of this work are as follows. First we develop some new geometric and topological structural properties of M2SNs. We show how these properties can be used as pruning tests by an algorithm for exactly solving the 2-connected Steiner network problem, using the GeoSteiner algorithmic framework developed in (19). Finally we report computational and experimental results using this algorithm which show the effectiveness of these new pruning tests and give new insight into when M2SNs are Hamiltonian cycles for randomly generated instances.

2 Preliminaries

2.1 Steiner trees and 2-connected Steiner networks

We begin by reviewing some basic properties of Steiner trees and 2-connected Steiner networks, and establishing our notation. A more comprehensive overview of Euclidean Steiner trees can be found in Chapter 1 of (2).

For both the *Steiner tree problem* and *2-connected Steiner network problem* we are given a finite set of points (known as *terminals*) with specified locations in the Euclidean plane and asked to find a network interconnecting the terminals of minimum total length. The network may contain vertices other than the terminals; these extra vertices are known as *Steiner points*. For the Steiner tree problem there is no constraint on the minimum network other than it be connected; it is easy to see that such a network must necessarily be a tree, referred to as a *minimum Steiner tree*. For the 2-connected Steiner network problem, the minimum network must also be 2-connected, meaning that the network continues to be an interconnection network for the given set of terminals if any edge of the network is deleted. Such a network is referred to as a *minimum 2-connected Steiner network* (M2SN).

This article is protected by copyright. All rights reserved.

Author Manuscript

Minimum Steiner trees have a number of useful geometric and topological properties. The most important of these are that: any two edges meeting at a vertex meet at an angle e $2\grave{A}/3$; every Steiner point has degree 3; and hence each pair of edges meeting at a Steiner point meet at an angle of exactly $2\grave{A}/3$. As mentioned in the introduction, a minimum Steiner tree is said to be *full* if each terminal has degree 1 in the tree. Such a tree is referred to as a *full Steiner tree* (FST). Note that if an FST has k terminals then it has exactly $k - 2$ Steiner points.

One of the most useful properties of M2SNs is the following decomposition property. Let N be an M2SN in a Euclidean plane for a given terminal set X. The edges of N can be uniquely partitioned into nonempty sets N_i such that each N_i is maximal with respect to the property that every distinct pair of edges in N_i can be connected by a path in N_i for which all interior vertices are Steiner points. We refer to each N_i as a *full component* of N. The number of terminals that a full component N_i interconnects is referred to as the *size* of N_i.

The following property is proved in (13).

Proposition 1. Each full component of an M2SN is an FST.

Note that an immediate consequence of Proposition 1 is that every node in an M2SN has degree 2 or 3. Furthermore, since the total degree sum of nodes in any network is even, it follows that the number of nodes of degree 3 in an M2SN must be even.

The decomposition of an M2SN into FSTs exactly parallels what happens in the Steiner tree problem; a minimum Steiner tree can also be viewed as a union of edge-disjoint FSTs. This implies that the GeoSteiner algorithm for solving the Steiner tree problem, discussed below in Section 2.2, can be adapted to solve the M2SN problem (19).

The following notation will be used throughout the remainder of this paper:

- A network N can be written as $N = (V, E)$, where V represents the set of vertices of N which are points in the plane, and E represents the set of edges of N which are line segments in the plane.
- A full component is said to be *non-trivial* if it contains at least one Steiner point, or equivalently has size ≥ 3.
- Throughout this paper we will use X to denote the given finite terminal set for the 2-connected Steiner network problem, and we denote the cardinality of X by $n = \text{card}(X)$.
- If N is a connected network, and N' is a connected subnetwork of N, then we use the notation $N \backslash N'$ to denote the network in which we delete all edges of N' and internal vertices of N' from N. All leaves of N' are considered to be vertices of

This article is protected by copyright. All rights reserved.

$N \backslash N'$ (even if they were not originally vertices of N). In other contexts '\' is used as the usual set difference operator.

2.2 GeoSteiner algorithm

The GeoSteiner algorithm is a general framework for solving Steiner tree problems in the plane to optimality (2). The main idea behind the algorithm is to first generate a set of candidate full components/full Steiner trees (FSTs), using pruning rules to keep this set down to a manageable size, and then find a minimum-length connected concatenation of these FSTs that spans all the given terminals. For the M2SN problem a similar strategy can be applied, but in this case different pruning rules are required for the generation phase, and a 2-connected network must be constructed in the concatenation phase.

FST generation in the GeoSteiner algorithm is based on a dynamic programming approach. Given any FST T and an edge pq in T, imagine that we cut edge pq at its midpoint. We obtain two *branch trees*: one rooted at p having an "unfinished" edge leaving p along pq, and another rooted at q having an "unfinished" edge leaving q along qp. We can think of a branch tree as an FST spanning a set of terminals and a single point at infinity. The set of all branch trees spanning the same given set of terminals with a given topology are referred to as a *branch*. In the Euclidean metric, the geometry of FSTs implies that all root candidates for a given branch must be located on a single subarc in such a way that the tree edges meeting at the root form angles of $2\pi/3$.

The generation phase of GeoSteiner constructs FSTs in a bottom-up manner by enumerating and merging branches of increasing size. For a set of n terminals, there are n branches of size 1, each corresponding to a terminal $u \in X$ with an infinite set of stems pointing in all possible directions. Branches and FSTs of size k are obtained by joining branches of size l with branches of size $k - l$, for $1 \le l \le \lfloor k/2 \rfloor$, in such a way that the geometric properties of a Euclidean Steiner tree are not violated.

Pruning tests for branches are based on reducing the feasible subarc of the root. The simplest is the *projection test* that basically uses the fact that edges meet at angles of $2\pi/3$ at Steiner points (2). The projection test remains valid for the 2-connected Steiner network problem. However, other pruning tests for the Euclidean Steiner tree problem, such as the lune test and the bottleneck Steiner distance test, are no longer valid once the extra constraint of 2-connectedness is applied.

The output of the FST generation phase is a set of FSTs $\mathcal{F} = \{T_1, T_2, \ldots, T_m\}$ that is guaranteed to contain the full components of at least one M2SN for X. In the FST concatenation problem, we need to identify a subset $\mathcal{F}^* \subseteq \mathcal{F}$ such that $\mathcal{F}^*$ forms a 2-connected network for X and has

This article is protected by copyright. All rights reserved.

Author Manuscript

minimum total length. The FST concatenation problem can be solved using integer programming and branch-and-cut — as described in the next section.

2.3 Construction of minimum 2-connected spanning networks

Consider the slightly simpler problem of constructing a minimum 2-connected *spanning* network on the given set of terminals X, that is, a 2-connected network with vertex set X of minimum total length. Hence, only terminal to terminal edges are allowed. Even in this simpler case the problem remains NP-hard over general metrics (13), and in this section we briefly discuss the construction of such networks using integer programming; see also (5; 14).

Formally, let $G = (X, \mathcal{E})$ be an undirected complete graph on X. Let the cost $c(e)$ e 0 of an edge $e = (u, v) \in \mathcal{E}$ be the Euclidean distance between terminals u and v (though, more generally, the model will work with any distance function). A network $\hat{N} = (X, \hat{E})$, $\hat{E} \subseteq \mathcal{E}$, is a minimum-length 2-connected network on X if $\hat{N}$ is 2-connected and $\sum_{e \in \hat{E}} c(e)$ is minimum among all such subnetworks in G. Note that a Hamiltonian cycle on X is always a feasible solution, and that the cost of a minimum-length Hamiltonian cycle is at most 4/3 times the cost of a minimum-length 2-connected spanning network when the edge-costs form a metric (14).

Like many other combinatorial optimisation problems, the minimum-length 2-connected spanning network problem can be modelled using (linear) integer programming. A spanning network can be represented using an incidence vector x, where $x_e = 1$ if edge $e \in \mathcal{E}$ is part of the spanning network, and otherwise $x_e = 0$. Clearly, the cost/length of a spanning network is then $\sum_{e \in \mathcal{E}} c(e) x_e$. For a set of terminals $W \subseteq X$, let $\delta(W) \subseteq \mathcal{E}$ denote the *cut* given by W, that is, the set of edges with exactly *one* endpoint in W. Then we have the following *cut formulation* for the minimum 2-connected spanning network problem:

$$\text{minimise} \quad \sum_{e \in \mathcal{E}} c(e) x_e \qquad [1]$$

$$\text{subject to} \quad \sum_{e \in \delta(W)} x_e \geq 2, \quad \varnothing \neq W \subset X \qquad [2]$$

$$x_e \in \{0,1\}, \quad e \in \mathcal{E}. \qquad [3]$$

As the number of cut constraints [2] is exponential in $|X|$, we apply branch-and-cut to the problem. The separation problem can be solved in polynomial time by solving at most $|X| - 1$ minimum cut (or maximum flow) problems.

Author Manuscript

This article is protected by copyright. All rights reserved.

A number of variants of the problem can be solved by performing minor modifications of the integer program — these problem variants are discussed in later sections of the paper:

- A minimum-length Hamiltonian cycle (or travelling salesman tour) can be computed by adding a degree 2 constraint for each terminal: $\sum_{e \in \delta(\{u\})} x_e = 2, u \in X$.

  Other degree constraints can similarly be enforced.
- Constraints on the number of edges in the network, e.g., a lower bound *L* and an upper *U*, can be enforced by adding the constraints $\sum_{e \in \mathcal{E}} x_e \geq L$ and $\sum_{e \in \mathcal{E}} x_e \leq U$.
- Finally, we can use the integer programming formulation to solve the FST concatenation problem for the 2-connected problem by defining $\mathcal{E}$ be a set of *hyperedges* in which each hyperedge corresponds to the set of terminals spanned by a candidate FST, and the cost of the hyperedge equals the length of the associated FST. In this case, the integer program can be solved using the branch-and-cut algorithm of Warme (17).

3 Structural Properties

In this section our focus is on structural properties that will aid in reducing the number of candidate full components in the generation phase. There are two types of structural properties that are important in this context. The first type consists of properties that can be exploited in a preprocessing phase aimed at bounding the maximum size (by which we mean maximum number of terminals) of a full component. The second type consists of properties that can be used to reduce the number of candidate full components either during or after the generation phase (but before concatenation).

3.1 Properties relevant to preprocessing

One of the most important contributions in (19) was the establishment of a lower bound for the length of an M2SN *N* spanning a given set of *n* terminals *X*, based on the assumption that it contains a full component with *k* or more Steiner points. It was shown that this lower bound increases with *k*. Once this lower bound exceeds a known upper bound for the problem instance, such as the length of the minimum 2-connected spanning network for the terminals, one obtains an upper bound on the size of full components in the M2SN. The effectiveness of this lower bound comes from using properties of the spanning network obtained by replacing every full component *T* of *N* by a minimum spanning tree on the terminals of *T*.

Author Manuscript

This article is protected by copyright. All rights reserved.

Author Manuscript

Here we show that this lower bound on the length of an M2SN can be improved by changing the conditions we place on N, resulting in a better bound on the maximum size of full components. We define N_k to be a minimum 2-connected Steiner network on X that happens to contain *exactly* $2k$ vertices of degree 3. (Note that if the points in X are in general position we would expect these degree 3 vertices to all be Steiner points, since any degree 3 terminal would have each pair of incident edges meeting at angles of $2\pi/3$. Nevertheless, we allow below for the possibility that some degree 3 points may be terminals.) Let $\overline{N}_k$ denote the network obtained by replacing each full component T in N_k by a minimum spanning tree for the set of terminals spanned by T. The network N_k satisfies the following properties:

Lemma 1. Let $\overline{N}_k$ be defined as above. Then

(i) $\overline{N}_k$ is 2-connected,

(ii) every cycle in $\overline{N}_k$ has at least 4 terminals, and

(iii) $\overline{N}_k$ has exactly $n + k$ edges.

This lemma closely resembles Lemma 6 of (19) and the proof is almost identical (after substituting N_k for N) even though the definition of N_k here is slightly different from that of N in (19). Hence, the proof is not included here. We simply note that the proofs of statements (i) and (ii) do not depend on the value of k, which is why the proofs from (19) still apply. Statement (iii) can be proved by a straightforward counting argument which results in an equality here (rather than the inequality in (19)) because of the fact that N_k contains exactly $2k$ vertices of degree 3.

For a given network $\overline{N}_k$ as defined above, we define an edge to be *simple edge* if it belongs to both N_k and $\overline{N}_k$, or, equivalently, is an edge between two terminals of N_k. Let S_k be the set of simple edges in $\overline{N}_k$, and let

$$s_k = \sum_{e \in S_k} |e|,$$

i.e., s_k is the total length of all simple edges in $\overline{N}_k$. Note that if $k > 0$ then the graph induced by the simple edges (X, S_k) is a forest since if any connected component of (X, S_k) were to contain a cycle then that cycle would include a terminal of degree 3 in N_k. Furthermore, each connected component of the forest (X, S_k) is either a path or an isolated terminal, since no terminals have degree 3 in (X, S_k). The importance of simple edges comes from the following result.

This article is protected by copyright. All rights reserved.

Lemma 2. Let N_k be an M2SN on a set of n terminals X such that $k > 0$, and let S_k be defined as above. Then

(i) card (S_k) ≥ $n - 3k$, and

(ii) there exists a set $S_k' \subseteq S_k$ such that card (S_k') = $n - 3k$ and the forest $F_k' := (X, S_k')$ consists of $3k$ connected components (where any isolated terminal with no incident edges is also considered to be a connected component).

Proof. Assume, initially, that all vertices of degree 3 in N are Steiner points. Let X_S denote the set of terminals in X that are adjacent to a Steiner point in N (that is, belong to a non-trivial full component of N). Let d_{FST} denote the sum of degrees of the elements of X_S in the forest of non-trivial full components of N. Since there are $2k$ Steiner points, each of degree 3, it follows that d_{FST} d $6k$. Hence, in the forest $F_k := (X, S_k)$, the sum of degrees of all elements of X is $2n - d_{\mathrm{FST}}$ e $2n - 6k$ (since every element of X has degree 2 in N). Thus card (S_k) = $n - d_{\mathrm{FST}}/2$ e $n - 3k$. Finally, if any terminals in X have degree 3 in N, then card (S_k) is greater than in the previous case, and a similar argument applies. This proves part (i) of the theorem.

For part (ii), note that since each connected component of F_k is either a path between two terminals or an isolated terminal, it follows that each connected component of F_k effectively makes a contribution of 2 to d_{FST}, and hence the number of such connected components is $d_{\mathrm{FST}}/2$. Furthermore, as indicated in previous paragraph, card (S_k) = $n - d_{\mathrm{FST}}/2$. Now, removing any edge from S_k increases the number of connected components of F_k by 1. Thus removing any $3k - d_{\mathrm{FST}}/2$ edges from S_k results in a set S_k' with cardinality $n - 3k$ such that (X, S_k') has $3k$ connected components, as required.

For any terminal set X and integer $k > 0$, Lemma 2 gives a simple way of computing an effective lower bound for s_k. The idea is to compute a forest $F = (X, E(F))$ with $n - 3k$ edges and $3k$ connected components that minimises $s_k' := \sum_{e \in E(F)} |e|$. This computation can be performed by solving an integer linear program as described in Section 2.3, where the edge and component cardinalities can be guaranteed by adding constraints requiring that the degree of each vertex is between 0 and 2 and that the number of edges is at least $n - 3k$. Clearly s_k' constitutes a lower bound for s_k.

Recall that the *Steiner ratio Á* is the supremum of the ratio of the total length of the minimum Steiner tree to the minimum spanning tree over all possible finite terminal sets in the Euclidean plane. It has long been conjectured that $\rho = \sqrt{3}/2$ (6); this conjecture is still open,

This article is protected by copyright. All rights reserved.

Author Manuscript

Author Manuscript

despite earlier claims of a proof (see (10) for more details). It has, however, been shown that the conjecture holds for any Steiner tree with no full component of size greater than 8 (11). More generally, the best known lower bound for $\acute{A}$ is $\acute{A}_0 = 0.82416874\ldots$, established by Chung and Graham (3).

In terms of the Steiner ratio, we have the following relationship between N_k and $\tilde{N}_k$:

$$(|\tilde{N}_k| - s_k)\rho + s_k \leq |N_k|. \quad [4]$$

Let UB_N be a (possibly heuristic) upper bound for $|N|$. For example, we could choose UB_N to be the length of a minimum 2-connected spanning network for X. Since we are trying to construct a network N_k that is a (global) M2SN, and which happens to contain exactly $2k$ vertices of degree 3, we can assume that $UB_N \in |N_k|$. Also, let M_k be a minimum 2-connected spanning network for X with exactly $n + k$ edges satisfying the three properties of Lemma 1. Then $|M_k| \leq |\tilde{N}_k|$. Using these bounds, and the lower bound s'_k for s_k discussed above, inequality [4] implies that

$$|M_k|\rho + s'_k(1-\rho) \leq UB_N. \quad [5]$$

This inequality can be rapidly checked for increasing values of k in order to find an upper bound for k. The approach is similar to that in (19), but considerably more effective due to the use of information about the simple edges of $\tilde{N}_k$. Once an upper bound for k is known, it follows from (8) that $k + 2$ is an upper bound for the size of any full component of N. In the algorithm, this bound can be computed as part of a preprocessing phase, and is then used during the generation phase to limit the size of any candidate full component.

3.2 Properties that can be used in FST generation

A useful property of full components that was proved in (9), but has not previously been included in any published algorithm for solving the M2SN problem, is the following.

Proposition 2. (Convex hull property) No full component of an M2SN for a terminal set X spans more than two terminals on the boundary of the convex hull of X. Furthermore, any full component that spans two terminals on the boundary of the convex hull of X does so on consecutive terminals on the boundary.

This property can easily be checked dynamically during the generation phase of GeoSteiner, meaning that it can not only reduce the number of FSTs generated, but also decrease the running time of the generation phase, by allowing some branches to be removed from consideration earlier.

This article is protected by copyright. All rights reserved.

Author Manuscript

One of the most effective pruning conditions in the generation phase of the GeoSteiner algorithm for computing minimum Steiner trees makes use of the 'lune property' (6), which states that for each edge e of a minimum Steiner tree T there is an open region, known as a *lune*, which does not contain any points of T other than the interior points of e. This property does not apply to M2SNs, due to the extra condition of 2-conectedness. The following result from (1), however, gives a condition for M2SNs that allows one to exclude a Steiner point (constructed during the generation phase) and its incident edges if certain inequalities hold.

Proposition 3. (3-lune property) Let a, b, c be three points in the plane and let T be a minimum Steiner tree on a, b, c. For a given M2SN N, the tree T cannot occur as a subnetwork of N if there exists a point p in $N \setminus T$ such that at least two of the following inequalities hold:

$$\begin{aligned} |T| &> |ap| + |bc|, \\ |T| &> |bp| + |ac|, \\ |T| &> |cp| + |ab|. \end{aligned} \quad [6]$$

The proof of this proposition appears in (1). The simplest way of using this result in GeoSteiner is as a method of eliminating full components of size 3 or more at the end of the generation phase. For example, for each size 3 full Steiner tree T generated, we can label the three terminals a, b, c and delete T from the list of candidate full components if there exists a terminal p such that two of the inequalities [6] hold. To eliminate a Steiner tree of size greater than 3, for each Steiner point we label the three neighbouring nodes a, b, c and again check the inequalities [6].

Geometrically, this is equivalent to checking whether a region known as a 3-lune contains elements of the terminal set X. Let T be an FST on $X_T = \{a, b, c\}$. Let D_a denote the open disc with centre a and radius $|T|-|bc|$; define D_b and D_c similarly. Then we define the *3-lune* for T to be the region $(D_a \;) \; D_b) \cup (D_a \cap D_c) \cup (D_b \cap D_c)$. In other words, the 3-lune is the union of the three intersections of pairs of the open discs. An example of a 3-lune is given in Figure 1.

Note that this method of eliminating FSTs does not reduce the generation time, but it can lead to a significant reduction of the time required for concatenation by reducing the size of the list of candidate full components.

In (1), it is shown that it is also possible to develop a dynamic pruning method based on Proposition 3 for reducing the time required for the generation phase of GeoSteiner. The details of the method can be found in that paper, where computational results indicating its

This article is protected by copyright. All rights reserved.

Author Manuscript

effectiveness are also included. We also note that this 3-lune pruning test is a specific example of more general k-lune tests which can be described for all $k \geq 3$. However, it is shown in (1) that the effectiveness of these tests diminishes rapidly as k increases, so in this paper we restrict our attention to $k = 3$.

The following theorem gives a useful length bound on full components of an M2SN.

Theorem 1. (FST length bound property) Suppose N is an M2SN on a given terminal set X. Let UB_N be an upper bound for $|N|$, let T be a full component of N, and let $T(X)$ be a minimum Steiner tree for X. Then $|T| \leq UB_N - |T(X)|$.

Proof. It is clear from (8) that $N \setminus T$ is connected, and interconnects X. Hence, by the definition of a minimum Steiner tree, $|T(X)| \leq |N \setminus T|$, and the result follows.

The power of this theorem lies in the fact that the length of a minimum Steiner tree $T(X)$ can be computed rapidly (say, as part of the preprocessing phase) using GeoSteiner, at least for uniformly distributed sets of terminals with up to several thousand elements. As discussed in the previous section, the preprocessing phase also computes an upper bound UB_N for $|N|$. During the generation phase we can keep track of the current length of any branch tree, and prune away any branch trees or FSTs for which this length exceeds $UB_N - |T(X)|$.

The next two theorems (2 and 3) make use of a key property of M2SNs from (8). Let H be an undirected connected graph. A *block* of H is defined to be a maximal 2-edge-connected subgraph of H. The *block-bridge tree* of H is defined to be the tree obtained by contracting each block to a vertex; and the *reduced block-bridge tree* is the connected graph obtained from a block-bridge tree by replacing each maximal simple path (with interior vertices of degree 2) by an edge. If N is an M2SN, and T is a full component of N, it is shown in (8) that the reduced block-bridge tree of $N \setminus T$ has the same topology (i.e., graph structure) as T, and furthermore that there is an isomorphism between them that preserves the leaves of T (which are, up to contraction, leaves of the reduced block-bridge tree of $N \setminus T$).

A consequence of this property is the following theorem, which can be used as a way of eliminating certain candidate full components at the end of the generation phase. Given a full Steiner tree T on terminal set X_T, we say that a pair of edges e and f of the complete graph on X_T are *T-path independent* if the paths in T connecting the endpoints of e and f, respectively, do not overlap. More generally, a set of three or more edges is said to be T-path independent if each pair of edges in the set is T-path independent.

Theorem 2. (Hamiltonian property) For a given non-trivial full Steiner tree T with terminal set X_T, suppose there exists a Hamiltonian cycle H on X_T such that if we remove some T-path independent

This article is protected by copyright. All rights reserved.

set of edges from H the total length of the remaining edges is less than $|T|$. Then T cannot be a subnetwork of any M2SN.

Proof. We begin by assuming that T satisfies the conditions of the theorem but is also a full component of an M2SN N, and argue by contradiction. (More generally, we could assume that T is any subnetwork of N, such as a strict subtree of a full component of N, and essentially the same argument would apply.) It is easy to check that the network $N2 := (N \setminus T) \cup H$, obtained by replacing the full component T of N by the Hamiltonian cycle H on X_T, is still 2-connected.

Now recall that a *chord* of a cycle C in a network is defined to be an edge of the network not in C but whose endpoints are vertices of C. We claim that all edges belonging to a T-path independent set of edges from H are chords of a cycle of $N2$ The justification of this claim is as follows. Because the reduced block-bridge tree of $N \setminus T$ has the same topology as that of T (8), we observe that for any set of T-path independent edges of H, say $E_{H,T}$, of cardinality l, there exists a set of l independent paths in $N \setminus T$ each of which connects the endpoints of one of the edges in $E_{H,T}$. Hence, the union of these paths with the edges of H not in $E_{H,T}$ forms a single cycle in $N2$, and each edge in $E_{H,T}$ is a chord of this cycle.

It is clear that no cycle in an M2SN has a chord, since any such chord can be removed without compromising the 2-connectedness of the network. Hence we can remove any T-path independent set of edges of H from $N2$and maintain 2-connectedness, contradicting the length-minimality of N.

This is the first known theorem to show that some full minimum Steiner trees can never occur as part of an M2SN. For example, consider a minimum Steiner tree T on the vertices of a unit square, as shown in Figure 2. For the Hamiltonian cycle running around the perimeter of the square, edges bc and ad are T-path independent, and the remaining two edges have a total length of 2. However it is easy to show that $|T| = 1 + \sqrt{3} > 2$, and hence it follows from Theorem 2 that T cannot be part of an M2SN.

Although this theorem is potentially very useful at removing candidate full components, it is inefficient to implement as part of GeoSteiner in full generality. For full components T of large size it can be time-consuming to find a Hamiltonian cycle satisfying the conditions of the theorem, if one exists (noting that the optimal Hamiltonian cycle required on the terminals of T is not necessarily one with minimum length). A simpler way of using this theorem is to only apply it to size 4 FSTs generated by GeoSteiner, and to size 4 subtrees of larger FSTs. In the latter case, for any FST of size ≥ 5 we consider any subtree T obtained by taking all edges incident to two adjacent Steiner points of the FST. These size 4 FSTs are easy to check: if the terminals of T are a, b, c, d, where a and d are adjacent to one Steiner

Author Manuscript

This article is protected by copyright. All rights reserved.

point, and b and c are adjacent to the other, then there are only two cases to consider – either $|(a, b)| + |(c, d)| < |T|$ or $|(a, c)| + |(b, d)| < |T|$ will invalidate T as a potential full component of an M2SN.

Initial computational experiments indicate that this is a very effective method of ruling out candidate full components after the generation phase of GeoSteiner. For 50 terminals randomly and uniformly distributed in a square just over 60% of all size 4 FSTs can be eliminated by applying this test after the generation phase. For the same instances, the (restricted) test eliminates about 84% of all size 5 FSTs, which is what we would expect (since size 5 FSTs contain two size 4 subtrees and $0.84 = 1 - (1-0.6)^2$). More computational details appear in Section 4.2.

3.3 Other structural properties

The next theorem again makes use of the concept of the block-bridge tree of a network. Given an M2SN N, we say that two full components of N, T_1 and T_2 are *mutually visible* if in the block-bridge tree corresponding to $N \setminus T_1$ (which we know is connected by (8)) no edge of T_2 is contracted (or, equivalently, no edge of T_2 belongs to a block of $N \setminus T_1$), and vice versa.

Theorem 3. (Mutually visible components property) No M2SN contains a pair of mutually visible full components each with size $2k$ for any integer $k \geq 2$.

Proof. Assume, contrary to the statement of the theorem, that there exists an M2SN N containing two mutually visible full components T_1 and T_2 each of size $2k$. By (8), the reduced block-bridge tree for $N \setminus T_1$ has the same topology as T_1; hence T_1 and T_2 must both have the same topology. Note that the block-bridge network for $N \setminus (T_1 \cup T_2)$ is a forest of $2k$ disjoint simple paths. Let F denote this forest of simple paths. See Figure 3 for an example where $k = 2$.

We can put a cyclic order $p_1, p_2, \ldots, p_{2k}$ on the terminals of T_1 corresponding to the order they are traversed in a counterclockwise outer walk of T_1 (where $p_{2k+1} := p_1$). Using this ordering, we then place a cyclic ordering $q_1, q_2, \ldots, q_{2k}$ on the terminals of T_2 such that each maximal path in F has endpoints p_i and q_i. Next we define Hamiltonian cycles H_1 for $\{p_1, p_2, \ldots, p_{2k}\}$ composed of edges $\{e_i := (p_i, p_{i+1}) : 1 \text{ d } i \text{ d } 2k\}$, and H_2 for $\{q_1, q_2, \ldots, q_{2k}\}$ composed of edges $\{f_i := (q_i, q_{i+1}) : 1 \text{ d } i \text{ d } 2k\}$. By construction, for each cycle H_j ($j = 1, 2$), $|H_j|$ is less than the total length of a complete outer walk of T_j, which traverses each edge of T_j twice; hence $|H_j|/2 < |T_j|$ (noting that because $k \text{ e } 2$ the inequality is strict). This implies that

$$\frac{|H_1| + |H_2|}{2} < |T_1| + |T_2|. \quad [7]$$

Author Manuscript

This article is protected by copyright. All rights reserved.

For $j = 1, 2$, let H_j^e be the set of edges of H_j with even index, and let H_j^o be the set of edges of H_j with odd index. Observe that $F \cup H_1^e \cup H_2^o$ forms a single cycle. Hence $(N \setminus (T_1 \cup T_2)) \cup H_1^e \cup H_2^o$ is a 2-connected Steiner network on the terminals of N. Similarly, $(N \setminus (T_1 \cup T_2)) \cup H_1^o \cup H_2^e$ is a 2-connected Steiner network on the terminals of N. But, since $H_j^e \cup H_j^o = H_j$ it follows that either $|H_1^e| + |H_2^o| \leq (|H_1| + |H_2|)/2$ or $|H_1^o| + |H_2^e| \leq (|H_1| + |H_2|)/2$, which by inequality [7] implies that either $|(N \setminus (T_1 \cup T_2)) \cup H_1^e \cup H_2^o|$ or $|(N \setminus (T_1 \cup T_2)) \cup H_1^o \cup H_2^e|$ is strictly less than $|N|$, contradicting the minimality of N.

This theorem has numerous interesting consequences, many of which are easy to check algorithmically. For example, it implies that it cannot be the case that the only full components of an M2SN N of size greater than 2 are two components of size 4 (as in Figure 3), since two such components would necessarily be mutually visible. Or more generally, if by the preprocessing properties discussed in Section 3.1 we know that N has at most $2k$ vertices of degree 3 where k is even, then N cannot contain more than one full component of size $k + 2$.

Theorem 3, or some of its consequences, could in theory be incorporated into the concatenation phase of the algorithm. However it seems unlikely that the extra constraints required in the integer program would result in any significant increase in the speed of concatenation. particularly for problems with less than 100 terminals. For this reason, the properties related to this theorem have not currently been implemented in the algorithm.

3.4 An interesting instance

We conclude this section with a problem instance that illustrates a couple of interesting negative properties of M2SNs, that help distinguish the structure of such networks from that of minimum Steiner trees. Two of the properties of Euclidean minimum Steiner trees that lead to the most effective pruning techniques are the lune property and the bottleneck Steiner distance property (see for example (18) or (2)). While neither of these properties apply to M2SNs, it seems possible that some of their consequences may be relevant to the 2-connected problem. In particular, the lune property implies that if a Euclidean minimum Steiner tree has a size 3 full component T_0, then there are no terminals in the interior of the convex hull of T_0. The bottleneck Steiner distance property implies that the longest edge in a Euclidean minimum Steiner tree is no longer than the longest edge in a minimum spanning tree on the same set of terminals; the equivalent result for 2-connected networks would be that the longest edge in an M2SN is no longer than the longest edge in a minimum 2-connected spanning network on the same terminals. While both these properties may seem

Author Manuscript

This article is protected by copyright. All rights reserved.

plausible, and indeed hold for all of the hundreds of randomly generated instances we investigated, neither is true. This is confirmed by the contrived instance shown in Figure 4.

Figure 4 shows an M2SN (containing two full components of size 3) on a set of 52 terminals. The corresponding minimum 2-connected spanning network is a Hamiltonian cycle indicated by the boundary of the lightly shaded region. The triangle Δabc, which is the convex hull of one of the full components of size 3, has several terminals in its interior. Furthermore, the longest edge in the M2SN is *sc* with length $|sc| = 8.56\ldots$, whereas the longest edge in the 2-connected spanning network is *pq* with length $|pq| = 7.12\ldots$. The figure also shows that terminals can be located very close to the boundary of the 3-lune of a full component (the dark shaded region), suggesting that the 3-lune is in some ways quite tight as a region in which terminals cannot occur.

4 Computational and Experimental Results

In this section we provide some more details on the algorithm and its implementation, and present computational results on the performance of each of the new pruning tests and the overall performance of the algorithm. We also examine some experimental results on applying the algorithm to randomly generated instances, looking in particular at the frequency of the occurrence of non-trivial full components in M2SNs.

4.1 The algorithm

As discussed earlier, the 2-connected version of GeoSteiner consists of three phases: a new *preprocessing phase*; a *generation phase*, which constructs a sufficient set of FSTs $\mathcal{F} = \{T_1, T_2, \ldots, T_m\}$ by efficient enumeration; and a *concatenation phase*, which identifies a subset $\mathcal{F}^* \subseteq \mathcal{F}$ such that the union of the elements of $\mathcal{F}^*$ interconnects *X*, is 2-connected and has minimum total length.

We indicate below the new steps in our algorithm.

- *Preprocessing phase:*
  1) We compute the convex hull of the set of terminals *X*.
  2) We compute a good upper bound UB_N on the length of an M2SN for *X*. In our implementation of the algorithm this is done by computing the length of a minimum 2-connected spanning network for *X*, as described in Section 2.3.
  3) We precompute a bound on *k*, the maximum number of Steiner points in any full component, by first computing the bound for *k* described in

Author Manuscript

This article is protected by copyright. All rights reserved.

(19), and then improving this bound by testing if inequality [5], from Section 3.1, holds for each k (see Section 3.2). Note that this inequality makes use of the bound UB_N from step (2). Observe also, that in order to find a valid upper bound for k we need to test inequality [5] for all values of k up to the bound in (19), as the left hand side of inequality [5] is not necessarily monotonically increasing with respect to k.

4) If the bound on k is greater than 0 in step (3), we run GeoSteiner on the terminal set X in order to compute $|T(X)|$, the length of a minimum Steiner tree $T(X)$ for X. This value and the bound UB_N from step (2) are used to compute the FST length bound $UB_N - |T(X)|$ (Theorem 1).

- *FST generation phase:* There are five new rules in the generation phase for speeding up the construction of the set of FSTs $\mathcal{F}$ and reducing its cardinality, based on the properties discussed in Sections 3.2 and 3.3.

1) We have a new bound on the size of any full component; this bound is 2 greater than the bound on k computed in preprocessing step (3) above. This is easily incorporated into the generation procedure to prevent the construction of any FST with size greater than the computed bound.

2) Using the convex hull information in step (1) of the preprocessing phase, we check that each branch and each candidate full component satisfies the convex hull property (Proposition 2).

3) We use the 3-lune property (Proposition 3) dynamically on the branches to rule out invalid families of full components as early as possible. We also check the property on all individual candidate full components as they are generated.

4) The FST length bound property (Theorem 1) gives an upper bound on the length of any full component, which is computed in step (4) of the preprocessing phase. This length bound is checked dynamically on the branches during the generation process.

5) Finally, the Hamiltonian property (Theorem 2) gives a test for eliminating certain candidate full components of size 3 or more. There is no obvious way of applying this test dynamically, so it is applied to all candidate full components at the end of the generation phase.

Author Manuscript

This article is protected by copyright. All rights reserved.

- *FST concatenation phase*. Optimal concatenation is achieved by solving the hypergraph version of the integer program described in Section 2.3.

4.2 Experimental setup

To assess the performance of each of the new pruning tests and the overall performance of the algorithm, we ran experiments on randomly-generated problem instances with uniformly-distributed terminals (with integer coordinates) inside a $10,000 \times 10,000$ square. The literature does not appear to contain any practical or benchmark instances specifically for the 2-connected Steiner network problem, and generating random uniform instances allows hundreds or thousands of individual seeds to be tested. We were, however, also able to apply the algorithm to selected instances of the library TSPLIB (16) for the Euclidean Travelling Salesman problem, where the vertices are given in terms of Cartesian coordinates.

With the exception of the FST size bound, the algorithm has been implemented in such a way that each of the new tests can be switched on and off, thereby allowing the individual impact of each test on algorithm performance to be observed. Although the new FST size bound cannot be switched off in the current implementation, the previous 'old' bound from (19) (as discussed in Section 3.1) is still reported as an output for comparison with the new bound.

The key outputs of interest are the number of FSTs remaining at the end of the FST generation phase, and CPU times for the generation and concatenation phases.

The experiments presented in this section were run on a desktop computer with an Intel Core i7-2600 CPU @ 3.40GHz x 8 processor and 7.8GB of memory.

4.3 Comparison of old and new FST size bounds

As we will see in Subsection 4.5, the FST size bound has a strong influence on algorithm performance. Figure 5 compares the FST size bounds from the implementation associated with (19) with the new FST size bounds; the comparisons have been run for uniform random problem instances with 25 to 50 terminals (in multiples of five). The average difference between the old and new bounds increases from 1.3 for 25 terminals to 2.4 for 50 terminals. This suggests that tighter bounds on the FST size would be likely to further improve algorithm performance and enable random instances beyond 50 terminals to be reliably solved.

4.4 Impact of the new pruning tests

Author Manuscript

This article is protected by copyright. All rights reserved.

Author Manuscript

Having compared the old and new FST size bounds, we now investigate the individual impact of the pruning tests on algorithm performance. For the remainder of this section, the *base case* will refer to the configuration in which the new FST size bound is enabled but all other new tests are disabled. Table 1 shows the impact of applying each new test individually to the base case, for problem instances with 25 terminals, with 100 seeds for each scenario. Our aim here is to assess the relative effectiveness of each of the pruning tests across the majority of random instances. For this reason we have only included results for random instances where the new maximum FST bound was at most five, as the less common instances with FST bound of six or more appeared to be skewing the results in unrepresentative ways (random instances with an FST bound of six or more accounted for 21% of the 100 seeds run in each scenario). In order to give an idea of the distribution of CPU times for the generation phase across the 100 seeds, the table includes bounds on the interquartile range (IQR), from the 25th to 75th percentile. Note that the preprocessing time has not been included in the table as it is insignificant when compared to the FST generation and concatenation times.

The table shows that the 3-lune test is the most powerful test for restricting the number of FSTs generated, while the FST length test is the most powerful test for reducing CPU time. In all cases CPU time for concatenation accounts for a small fraction of total CPU time. The unintuitive reduction in FST generation time for the Hamiltonian test (which is performed at the very end of the FST generation phase) is due to reductions in internal book keeping and time costs associated with setting up data structures for FSTs in a form that can be used in the concatenation phase. Note that the potential effectiveness of the Hamiltonian test at eliminating FSTs may be under-represented since the current implementation of the Hamiltonian test is only applied to size-4 FSTs and size-4 sub-components of larger FSTs. It is also worth mentioning that the potential effectiveness of the Hamiltonian test increases with increasing problem size, since larger problems will generally have more larger-sized FSTs with size-4 subcomponents.

Table 2 shows the impact of applying each new test cumulatively starting with the base case, and adding one test at a time, for 100 seeds of problem instances with 25 terminals. The order in which the tests are applied has been selected so that tests which require less processing time are applied ahead of those requiring more processing time. For random instances with 25 terminals, the combined new pruning tests, when compared to the base case, provide an average reduction of 93% in the number of FSTs generated and an average reduction of 77% to the CPU time.

4.5 Performance of the new algorithm based on instance size

This article is protected by copyright. All rights reserved.

Author Manuscript

Table 3 summarises the algorithm performance as the number of terminals is increased, for the case when all of the new tests are enabled. In contrast to the previous sections, all randomly generated instances are solved; there is no artificial restriction placed on the maximum FST bound.

As the number of terminals increases, the number of FSTs generated shows, at worst, quadratic growth , while the total CPU time (which continues to be dominated by FST generation time) appears to grow exponentially. The maximum total CPU time required for any of the seeds in the case of 50 terminals was approximately 5.9 hours. Experiments beyond 50 terminals cannot be reliably solved by the current algorithm due to excessive CPU times.

Figure 6 shows the relationship between the FST size bound and the number of surviving FSTs (left) or FST generation time (right). The results in the figure are for combined output data relating to the 600 problem presented in Table 3, with 25 to 50 terminals. The figure shows that the number of FSTs generated increases at approximately a quadratic rate with respect to the FST size bound, while FST generation CPU time displays a close to exponential increase. Note that these growth rates roughly correlate with those seen in Table 3, suggesting that the increases in FSTs and P1 CPU time resulting from increasing the number of terminals is largely driven by the associated increase in the FST size bound.

A final observation about the FST size bound relates to rectangular lattice problem instances in which the set of terminals is arranged in an $m \times n$ regular lattice of unit squares. It can be shown that the new FST size bound for rectangular lattice instances is always exactly two, and experiments confirm that the new algorithm solves square lattice instances very quickly, with problem sizes up to 64 terminals being solved within seconds. This is in contrast to the Euclidean Steiner tree problem, for which rectangular lattice problems typically take longer to solve than random problem instances, because of a reduction in pruning efficiency due to the increased frequency of large full components (see also (13)).

4.6 Frequency of non-Hamiltonian solutions

It was observed in (19) that small instances of the M2SN problem very often have solutions that are also solutions to the travelling salesman problem for the given set of terminals, i.e., such instances have Hamiltonian solutions. We have further investigated the frequency with which M2SNs are Hamiltonian by running experiments in which the FST size bound has been manually specified. When the FST size bound is set to two, the program returns a solution to the minimum 2-connected spanning network problem, and if the FST size bound is restricted to three, the program returns a minimum 2-connected network interconnecting the terminals in which each Steiner point is adjacent to three terminals. In both cases, if the

This article is protected by copyright. All rights reserved.

Author Manuscript

solution is non-Hamiltonian then the M2SN on the same set of terminals is also non-Hamiltonian. Restricting the FST size bound manually allows problem instances with more than 50 terminals to be run. Figure 7 shows the proportion of non-Hamiltonian solutions for 1000 seeds (for each cardinality of the terminal set) as the number of terminals is increased (in multiples of 10) from 10 to 100, when the maximum FST size is manually restricted to two and three.

Preliminary experiments with between 100 and 200 terminals indicate that the percentage of non-Hamiltonian solutions continues to increase at an almost linear rate with increasing problem size, with approximately 70% of solutions being non-Hamiltonian for 200 terminals. Clearly, however, this rate of increase will eventually decrease, since the percentage of non-Hamiltonian solutions is bounded above. The key observation is that for large problem instances a Hamiltonian solution will be the exception rather than the rule.

In addition to uniform random instances, we also examined problem instances for which terminals are distributed according to capped Gaussian distributions with respect to the x and y coordinates; these tests provided similar results to those shown in Figure 7. A third set of experiments was undertaken in which the terminals are distributed according to a set of four Gaussian clusters with one cluster in each quadrant. For these cases the proportion of non-Hamiltonian solutions was significantly lower than for the uniform and single-cluster Gaussian problem instances.

4.7 Special problem instances

To continue to build intuition about the structure of M2SNs, a number of contrived and practical problem instances with interesting properties were investigated. Using the new algorithm it was determined that problem instances ‘att48’ (with 48 terminals) and ‘berlin52’ (with 52 terminals) from the TSPLIB database (16) have Hamiltonian solutions. A variety of square and rectangular lattice problems were also determined to have Hamiltonian solutions, confirming results from (13).

Finally, Figure 8 shows a number of special instances with interesting properties. Example (a) from (19) verifies that non-Hamiltonian solutions are possible for problem instances with as few as eight terminals. (It is conjectured that problem instances with less than 8 terminals always have Hamiltonian solutions, but this has only been proved if the number of terminals is less that 6.) Example (b) has a non-Hamiltonian solution with a size-4 FST and two size-3 FSTs, and is the smallest known instance containing a size-4 full component. Example (c) is the smallest known non-basic solution (a non-basic 2-connected network in the plane has the property that when the outer cycle is removed, the remaining network is not a forest).

This article is protected by copyright. All rights reserved.

Author Manuscript

5 Conclusion

We have presented a new exact algorithm for computing minimum 2-connected Steiner networks in the Euclidean plane. The algorithm is based on the GeoSteiner framework for computing minimum Steiner trees in the plane. Several new geometric and topological properties of minimum 2-connected Steiner networks have been developed and incorporated into the new algorithm. Experimental results show that for random problem instances with 25 terminals, the new algorithm provides an average reduction of 93% in the number of FSTs generated and an average reduction of 77% to the CPU time, when compared to a reference base case in which all but one of the new pruning tests is disabled. The new algorithm can reliably compute exact solutions to randomly generated instances with up to 50 terminals - doubling the range of existing exact algorithms. Further experiments also demonstrated that, up to 200 terminals, the appearance of non-Hamiltonian solutions to the minimum 2-connected Steiner network problem increases approximately linearly with increasing problem size.

References

[1] M. Brazil, M. Volz, M. Zacharaisen, C.Ras, and D. Thomas, *New pruning rules for the Steiner tree problem and 2-connected Steiner network problem*, submitted, 2017.

[2] M. Brazil and M. Zacharaisen, *Optimal Interconnection Trees in the Plane*, Springer International, Switzerland, 2015.

[3] F.R.K. Chung and R.L. Graham, *A new bound for Euclidean Steiner minimal trees*, Ann. New York Acad. Sci. **440** (1985), 328-346.

[4] V.G. Deineko, R. van Dal and G. Rote, *The convex-hull-and-line Traveling Salesman Problem: A solvable case*, Inform. Process. Lett. **51** (1994), 141-148.

[5] G.A. Dirac, *Minimally 2-connected graphs*, J. Reine Angew. Math. **228** (1967), 204-216.

[6] E.N. Gilbert and H.O. Pollak, *Steiner minimal trees*, SIAM J. Appl. Math. **16** (1968), 1-29.

[7] D.F. Hsu and X.-D. Hu, *On shortest two-edge connected Steiner networks with Euclidean distance*, Networks **32** (1998), 133-140.

[8] K. Hvam, L. Reinhardt, P. Winter, and M. Zachariasen, *Bounding component sizes of two-connected Steiner networks*, Inform. Process. Lett. **104** (2007), 159-163.

This article is protected by copyright. All rights reserved.

Author Manuscript

[9] K. Hvam, L. Reinhardt, P. Winter, and M. Zachariasen, *Some structural and geometric properties of two-connected Steiner networks*, Computing: The Australasian Theory Symposium (CATS2007), Ballarat, Australia, Conferences in Research and Practice in Information Technology (CRPIT), vol. 65, 2007, pp. 85-90.

[10] A.O. Ivanov and A.A. Tuzhilin, *The Steiner ratio Gilbert-Pollak conjecture is still open*, Algorithmica **62** (2012), 630-632.

[11] D. Kirszenblat, *The Steiner ratio conjecture for eight points*, Master's thesis, Univ. of Melbourne, 2014.

[12] E.L. Luebke, *K-connected Steiner network problems*, Ph.D. thesis, Univ. of North Carolina, 2002.

[13] E.L. Luebke and J.S. Provan, *On the structure and complexity of the 2-connected Steiner network problem in the plane*, Oper. Res. Lett. **26** (2000), 111-116.

[14] C.L. Monma, B.S. Munson, and W.R. Pulleyblank, *Minmum-weight two-connected spanning networks*, Math. Program. **46** (1990), 153-171.

[15] S. Peng, M. Li, S. Zhang, and T.C. Edwin Cheng, *Some new structural properties of shortest 2-connected Steiner networks*, F.P. Preparata and Q. Fang (Eds.): FAW 2007, LNCS 4613, 2007, pp. 317-324.

[16] G. Reinelt, *TSPLIB-A traveling salesman problem library*, ORSA J. Comput. **3** (1991), 376-384.

[17] D.M. Warme, *Spanning trees in hypergraphs with applications to Steiner trees*, Ph.D. thesis, Univ. of Virginia, 1998.

[18] D. M. Warme, P. Winter, and M. Zachariasen, *Exact algorithms for plane Steiner tree problems: A computational study*, in D.-Z. Du, J. M. Smith, and J. H. Rubinstein (eds.): Advances in Steiner Trees, Kluwer Academic Publishers, Boston, 2000, pp. 81-116.

[19] P. Winter and M. Zachariasen, *Two-connected Steiner networks: Structural properties*, Oper. Res. Lett. **33** (2005), 395-402.

Figure 1 The Steiner minimum tree on *a*, *b*, *c* cannot be a full component of an M2SN if there are any terminals of the M2SN in the shaded region.

Figure 2 By Theorem 2, this Steiner tree on the vertices of a unit square cannot be part of an M2SN.

This article is protected by copyright. All rights reserved.

Author Manuscript

Figure 3 Two mutually visible size 4 full components, T_1 and T_2, of a 2-connected Steiner network, and the block-bridge forest for the remainder of the network. The total length of the network can be reduced (while maintaining 2-connectedness) by replacing T_1 and T_2 by either $\{e_2, e_4, f_1, f_3\}$ or $\{e_1, e_3, f_2, f_4\}$.

Figure 4 An instance on 52 terminals, illustrating a number of properties of M2SNs.

Figure 5 Comparison of maximum FST size bounds for the problem instances in Table 3. The old and new curves represent averages across 100 seeds. The horizontal bars indicate the relative frequency distribution for the FST size bound, with the distribution for the old bound on the left and that for the new bound on the right. The upper bound is given by $\lfloor n/3 \rfloor + 1$ where n is the number of terminals; this upper bound was established in (8).

Figure 6 Boxplots showing the relationship between the FST size bound with (left) FSTs generated and (right) FST generation time for the problem instances in Table 3 (600 seeds in total). Lower, middle and upper hinges correspond to the 25th, 50th and 75th percentiles respectively. The upper and lower whiskers extend from the hinges to the largest or smallest (respectively) values no further than 1.5 times the inter-quartile range. Outliers beyond the ends of the whiskers are plotted individually. Data for FST size bounds less than four and greater than nine have been removed due to insufficient sample sizes. Note that the boxplot on the right uses a log scale.

Figure 7 The percentage of non-Hamiltonian solutions for problem instances with 10 to 100 terminals for 1000 seeds (for each cardinality of the terminal set), when the maximum FST size is restricted to two and three.

Figure 8 A selection of special problem instances: (a) an 8-terminal instance with a non-Hamiltonian solution (19); (b) a 15-terminal instance with a size-4 FST; (c) a 16-terminal instance with a non-basic solution.

Table 1 Impact of individual tests on algorithm performance for 100 random problem instances with 25 terminals. The base case has the new FST bound enabled, but all other new tests disabled, and each test is applied individually to the base case. Column 2 shows the average number of candidate FSTs at the end of the generation phase (P1). Column 3 is the average CPU time for the P1 phase. Column 4 shows the bounds on the interquartile range (IQR), from the 25th to 75th percentile. The final two columns show the CPU time for the concatenation phase (P2), and the total running time for P1 and P2.

Test	**FSTs**	**P1 CPU (ms)**	**P1 IQR (ms)**	**P2 CPU (ms)**	**Tot CPU (ms)**
Base case	7101	602	[100, 1040]	93	695

This article is protected by copyright. All rights reserved.

Convex hull test	4665	418	[80, 730]	52	470
FST length bound	1565	146	[40, 230]	17	163
3-lune test	847	591	[120, 1005]	11	602
Hamiltonian test	3010	595	[100, 1030]	31	626

Table 2 Cumulative impact of new tests on algorithm performance for 100 random problem instances with 25 terminals. The base case has the new FST bound enabled, but all other new tests disabled, and each test is applied additively to the previous case. The interpretations of Columns 2 to 6 are the same as for Table 1.

Test	**FSTs**	**P1 CPU (ms)**	**P1 IQR (ms)**	**P2 CPU (ms)**	**Tot CPU (ms)**
Base case	7101	602	[100, 1040]	93	695
+ convex hull test	4665	418	[80, 730]	52	470
+ FST length bound	1472	150	[40, 235]	16	166
+ 3-lune test	586	157	[40, 250]	6	163
+ Hamiltonian test	504	152	[40, 240]	8	160

Table 3 The impact of increasing the number of terminals with all new tests enabled. The results in each row are for 100 random problem instances with the given cardinality of the terminal set. The interpretations of Columns 2 to 6 are the same as for Tables 1 and 2.

Terminals	**FSTs**	**P1 CPU (ms)**	**P1 IQR (ms)**	**P2 CPU (ms)**	**Tot CPU (ms)**
25	518	353	[50, 350]	9	362
30	802	2006	[268, 1925]	20	2026
35	1161	13,692	[1150, 13,612]	38	13,730
40	1564	75,253	[4552, 60,078]	64	75,317
45	2075	504,933	[19,318, 413,238]	102	505,035
50	2616	1,856,787	[117,925, 2,565,590]	184	1,856,971

Author Manuscript

This article is protected by copyright. All rights reserved.