Speeding up the Dreyfus-Wagner Algorithm for minimum Steiner trees

Bernhard Fuchs*
Center for Applied Computer Science Cologne (ZAIK)
University of Cologne, Weyertal 80, 50931 Köln, Germany

Walter Kern† Xinhui Wang‡
University of Twente, Department of Applied Mathematics
Faculty of EEMCS, P.O.Box 217
7500 AE Enschede, The Netherlands

Abstract

The Dreyfus-Wagner algorithm is a well-known dynamic programming method for computing minimum Steiner trees in general weighted graphs in time $O^*(3^k)$, where k is the number of terminal nodes to be connected. We improve its running time to $O^*(2.684^k)$ by showing that the optimum Steiner tree T can be partitioned into $T = T_1 \cup T_2 \cup T_3$ in a certain way such that each T_i is a minimum Steiner tree in a suitable contracted graph G_i with less than $\frac{k}{2}$ terminals. In the rectilinear case, there exists a variant of the dynamic programming method that runs in $O^*(2.386^k)$. In this case, our splitting technique yields an improvement to $O^*(2.335^k)$.

1 Introduction

The *Steiner tree problem* is one of the most well-known NP-hard problems: Given a graph $G = (V, E)$ of order $n = |V|$, edge costs $c \in \mathbb{R}_+^E$ and a set

*Email: bfuchs@zpr.uni-koeln.de
†Email: kern@math.utwente.nl
‡Email: xinhuiw@math.utwente.nl

$S \subseteq V$ of $k = |S|$ *terminal nodes*, we are to find a minimum cost subtree $T = T(S)$ of G connecting (spanning) all terminal nodes. Obviously, we may assume *w.l.o.g.* that C satisfies the triangle inequality and G is a complete graph (define edge costs by shortest paths).

The Steiner tree problem has been investigated extensively with respect to approximation (for a recent survey, see [1]) and computational complexity, both from a theoretical and practical point of view, *cf.*, *e.g.*, [3] for an overview and [7]. Particular attention has been paid to the *rectilinear Steiner tree problem*, *i.e.*, the case where the graph is a grid graph in the plane. For this case, which remains NP-complete [5], so-called exact algorithms have been designed [3], solving the problem in

$$O^*(2.386^k) = O(2.386^k poly(n))$$

(Here and in the sequel we adopt the O^*-notation to indicate that polynomial factors, *i.e.* factors of order $O(poly(n))$ are suppressed.)

The goal of this paper is to present a modification of the well-known Dreyfus-Wagner algorithm(*cf.* [2] or section 2). In addition, the worst case complexity of $O^*(3^k)$ of the Dreyfus-Wagner algorithm is – as far as we know – currently still the best for solving the problem in general graphs. We shortly describe the Dreyfus-Wagner algorithm in section 2. Section 3 then presents our modification, yielding an improved worst case complexity of order $O^*(2.684^k)$. In section 4, Fößmeier and Kaufmann's algorithm is slightly modified and used as a subroutine in our algorithm to obtain a runtime of $O^*(2.335^k)$ for the rectilinear case.

2 The Dreyfus-Wagner Algorithm

The Dreyfus-Wagner algorithm solves the Steiner tree problem for $S \subseteq V$ by dynamic programming. More precisely, it computes optimal trees $T(X \cup v)$ for all $X \subseteq S$ and $v \in V$ recursively.

The crucial observation is as follows. Assume first that v is a leaf of the (unknown) optimal tree $T(X \cup v)$. Then v is joined in $T(X \cup v)$ to some node w of $T(X \cup v)$ along a shortest path P_{vw}, such that either $w \in X$ or $w \notin X$, *i.e.*, w is a Steiner node in $T(X \cup v)$. In both cases we have $T(X \cup v) = P_{vw} \cup T(X \cup w)$. In case w is a Steiner node, it splits $T(X \cup w)$, *i.e.*, we can decompose $T(X \cup w) = T(X' \cup w) \cup T(X'' \cup w)$ for some nontrivial

bipartition $X = X' \cup X''$. We may thus write (abusing the notation slightly in an obvious way)

$$T(X \cup v) = \quad \min \quad P_{vw} \cup T(X' \cup w) \cup T(X'' \cup w), \tag{1}$$

where the minimum is taken over all $w \in V$ and all nontrivial bipartitions $X = X' \cup X''$. Note that (1) also holds in case $w \in X$ if we let $X' = X \setminus \{w\}$ and $X'' = \{w\}$. Finally, note that (1) also remains valid without our assumption of v being a leaf in $T(X \cup v)$. Indeed, if v is an internal node of $T(X \cup v)$, we may simply take $w = v$ (and $P_{vw} = \varnothing$).

The recursion (1) thus allows us to compute all optimal trees $T(X \cup v)$ for $v \subseteq V$ and $X \subseteq S$ of size $|X| = i$ recursively for $i = 1, 2, \cdots, k$. Assuming that we have already computed all these trees up to level $i-1$, the minimum in (1) for a given $X \subseteq S$ of size $|X| = i$ can be computed in time $O^*(2^i)$. Hence, in total the algorithm takes

$$O^*(\sum_{i=1}^{k} \binom{k}{i} 2^i) = O^*(3^k).$$

3 Improving the Dreyfus-Wagner Algorithm

The basic idea for improvement is as follows. We use the Dreyfus-Wagner algorithm to compute minimum Steiner tree for all subsets of S of size at most $\frac{k}{2}$ (or even less), and then seek to compose the minimum Steiner tree for S from these smaller trees. The basic difficulty to overcome is the following. Assume we knew that the minimum Steiner tree T for S contains some point v whose removal splits T into three branches T', T'' and T''', connecting three corresponding subsets S', S'' and S''' of S of size approximately $\frac{k}{3}$ each. Then v is the unique node splitting T into components of size at most $\frac{k}{2}$. On the other hand, exhaustive search for all possible (de-) compositions of T into three such subtrees amounts to search for all partitions $S = S' \cup S'' \cup S'''$ into sets of size $\frac{k}{3}$ (and the unknown node v). The time needed by such an exhaustive search would be

$$\binom{k}{k/3}\binom{2k/3}{k/3} \approx 3^k,$$

due to Stirling's formula.

For this reason, the standard way of decomposing T (as in the Dreyfus-Wagner algorithm) turns out to be inadequate. We use instead the following kind of decomposition.

Definition. An *r-split* of a tree $T \subseteq E$ is a partition

$$T = T_1 \cup \cdots \cup T_r$$

such that each

$$T_1 \cup \cdots \cup T_i, \qquad i = 1, \ldots, r$$

is connected.

Now consider a fixed minimum Steiner tree T for $S \subseteq V$ and an $r-$split $T = T_1 \cup \cdots \cup T_r$ as above. So, T_1 is a subtree of T and for $i \geq 2$, $T_i \subseteq E$ consists of several components, each of them containing exactly one node in the set $V(T_i) \cap [V(T_1) \cup \cdots \cup V(T_{i-1})]$. More precisely, let us define

$$\begin{array}{ll} A_i^- := V(T_i) \cap [V(T_1) \cup \cdots \cup V(T_{i-1})] & \\ A_i^+ := V(T_i) \cap [V(T_{i+1}) \cup \cdots \cup V(T_r)] \setminus A_i^- & \\ A_i := A_i^+ \cup A_i^- & (i = 1, \ldots, r) \\ S_i := S \cap V(T_i) \setminus A_i & \end{array} \tag{2}$$

We refer to $A := A_1 \cup \cdots \cup A_r$ as the set of *split nodes*. A split node $a \in A_i^-$ connects a component of T_i to $T_1 \cup \cdots \cup T_{i-1}$, while $a \in A_i^+$ is good for connecting a component of some $T_j, j > i$ to T_i. The sets $S_i, i = 1, \ldots, n$ are pairwise disjoint and if $|A|$ is "small" compared to $k = |S|$, the S_i are close to forming a partition of S. Using this kind of split-decomposition, it can be shown that T has a 2-split with $|S_1|, |S_2| \approx \frac{k}{2}$. As we will see, a (theoretically) even faster algorithm is obtained by considering certain 3-splits of T. Before analyzing these in detail, however, let us first state some simple facts.

Recall that we assume G to be complete. For $B \subseteq V$, we denote G/B the graph obtained from G by identifying all vertices $b \in B$ with a new vertex v_B (*i.e.*, contracting all the $|B|(|B|-1)/2$ edges induced by B). Edge costs in G/B are again defined via shortest path distances.

Lemma 1 Let $T \subseteq E$ be a minimum Steiner tree for $S \subseteq V$ and let $T = T_1 \cup \cdots \cup T_r$ be an $r-$split. Let $A_i^{\pm}$ and S_i be defined as in (2). Then
(i) $T_1 \cup \cdots \cup T_i$ is a minimum Steiner tree for $S_1 \cup \cdots \cup S_i \cup A_1 \cup \cdots \cup A_i$.
(ii) T_i is a minimum Steiner tree for $S_i \cup A_i^+ \cup v_{A_i^-}$ in G/A_i^-.

Proof. (i) Let $\tilde{T} \subseteq E$ be any tree connecting $S_1 \cup \cdots \cup S_i \cup A_1 \cup \cdots \cup A_i$ in G. Then it is straightforward from the definition of $r-$split that

$$\tilde{T} \cup T_{i+1} \cup \cdots \cup T_r$$

connects $S_1 \cup \cdots \cup S_r \cup A$. But $T = T_1 \cup \cdots \cup T_r$ is a minimum Steiner tree connecting $S_1 \cup \cdots \cup S_r \cup A$, implying

$$c(T_1 \cup \cdots \cup T_r) \leq c(\tilde{T} \cup T_{i+1} \cup \cdots \cup T_r)$$

Hence $c(T_1 \cup \cdots \cup T_i) \leq c(\tilde{T})$, proving (i).

(ii) Each component of T_i is joined to $T_1 \cup \cdots \cup T_{i-1}$ by a (unique) common point in A_i^-. Therefore, T_i is a tree in G/A_i^-. Furthermore, A_i^+ is, by definition, disjoint from A_i^- and spanned by T_i. Summarizing, T_i is a Steiner tree for $S_i \cup A_i^+ \cup v_{A_i^-}$ in G/A_i^-.

We are left to prove minimality of T_i. Let $\tilde{T}_i \subseteq E$ be any Steiner tree for $S_i \cup A_i^+ \cup v_{A_i^-}$ in G/A_i^-. Then certainly

$$T_1 \cup \cdots \cup T_{i-1} \cup \tilde{T}_i \subseteq E$$

is connected (as $\tilde{T}_i$ connects to v_{A_i}) and spans

$$S_1 \cup \cdots \cup S_i \cup A_1 \cup \cdots \cup A_{i-1} \cup A_i^+ = S_1 \cup \cdots \cup S_i \cup A_1 \cup \cdots \cup A_i$$

(as $A_i^- \subseteq A_1 \cup \cdots \cup A_{i-1}$).

Hence we conclude from (i) that $c(T_i) \leq c(\tilde{T}_i)$, proving the minimality of T_i. □

In what follows, we focus on 3-splits of the minimum Steiner tree T for $S \subseteq V$, $|S| = k$. Note that any 3-split $T = T_1 \cup T_2 \cup T_3$ may also be considered as a 2-split $T = T_1 \cup (T_2 \cup T_3)$. So, in particular, the following result implies the existence of a 2-split $T = T_1 \cup T_2$ with $|S_1| \approx |S_2| \approx \frac{1}{2}k$.

Theorem 2 For each $\epsilon > 0$ there exists a number $M = M_\epsilon = O(\frac{1}{\epsilon})$ such that the following holds: Any minimum Steiner tree T for $S \subseteq V$ with $k = |S|$ large enough allows a 3-split $T = T_1 \cup T_2 \cup T_3$ with $|S_1|, |S_2| \leq (\alpha + \epsilon)k$, $|S_3| \leq (1 - 2\alpha + 2\epsilon)k$ and $|A| \leq M$ for any prescribed value of $\alpha \leq \frac{1}{2}$.

Remark 3 The bound $M = O(\frac{1}{\epsilon})$ can probably be improved to $M = O(\log \frac{1}{\epsilon})$. For our purposes, however, it suffices to know that for each fixed $\epsilon > 0$ there is a constant upper bound on the size of $|A|$.

Proof of Theorem 2: There exists a vertex $v \in V(T)$ such that each component of $T \backslash v$ connects at most $\frac{k}{2}$ elements from S. Each such component connecting more than $\frac{k}{4}$ elements from S can again be split in a similar way. Continuing this process, we exhibit a set $A_0 \subseteq V(T)$ of size at most $M = \lceil \epsilon^{-1} \rceil$ such that each component C_j of $T \setminus A_0$ is "small" in the sense that it connects a subset $S_j^0 \subseteq S$ of at most ϵk elements.

We now construct T_1 by successively adding such small components, one at a time, together with the corresponding connecting nodes $a \in A_0$, say,

$$V(T_1) = V(C_1) \cup \cdots \cup V(C_s) \cup A_0', \qquad A_0' \subseteq A_0$$

until

$$(\alpha - \epsilon)k \leq \sum_{j=1}^{s} |S_j^0| \leq \alpha k$$

holds. We then construct T_2 by extending T_1 in a similar way, *i.e.*,

$$V(T_1 \cup T_2) = V(C_1) \cup \cdots \cup V(C_s) \cup \cdots \cup V(C_t) \cup A_0' \cup A_0''$$

with

$$(\alpha - \epsilon)k \leq \sum_{j=s+1}^{t} |S_j^0| \leq \alpha k$$

and set $T_3 := T \setminus (T_1 \cup T_2)$.

Note that, by construction, $T = T_1 \cup T_2 \cup T_3$ is a 3-split with corresponding set of split nodes $A \subseteq A_0$. The set S_1 consists of all $S_j^0, j = 1, \cdots, s$, plus the set $S \cap A_0' \setminus A$. Hence

$$(\alpha - \epsilon)k \leq |S_1| \leq \alpha k + |A_0| \leq (\alpha + \epsilon)k$$

for all $k \geq \frac{M}{\epsilon} = \frac{1}{\epsilon^2}$. Similarly,

$$(\alpha - \epsilon)k \leq |S_2| \leq (\alpha + \epsilon)k$$

holds, establishing the proof of the claim. □

The algorithm. After these preliminaries, it should now be clear how to proceed. Given $\epsilon > 0$ and a suitable $\alpha \leq \frac{1}{2}$ (to be determined below), we apply the Dreyfus-Wagner algorithm to compute minimum Steiner trees for all subsets of type

$$\widetilde{S} \cup \widetilde{A}^+ \cup v_{\widetilde{A}^-}, \qquad \widetilde{S} \subseteq S, \qquad |\widetilde{S}| \leq (\alpha + \epsilon)k$$

in $G/\widetilde{A}^-$, for all disjoint subsets, $\widetilde{A}^+, \widetilde{A}^- \subseteq A$ and all $A \subseteq V$ of size at most M. The number of possible choices for $\widetilde{A}^+$ and $\widetilde{A}^-$ is bounded by n^M, which is polynomial in n. Assuming that k is large enough, we may assume that

$M \leq \epsilon k$, so that $|S \cup A| \leq (1+\epsilon)k$ and, similarly, $|\widetilde{S} \cup \widetilde{A}^+| < (\alpha + 2\epsilon)k$. So this computation takes

$$2^{2M} \sum_{i=2}^{(\alpha+2\epsilon)k} \binom{(1+\epsilon)k}{i} 2^i = O^*\binom{(1+\epsilon)k}{(\alpha+2\epsilon)k} 2^{(\alpha+2\epsilon)k}$$

in total.

The second part of the algorithm is an exhaustive search for the 3−split $T = T_1 \cup T_2 \cup T_3$ whose existence is assured in Theorem 3.2. Basically, this comes down to finding the associated sets S_i (plus the corresponding sets of split nodes A_i out of a polynomial number of possible choices). For a fixed set of split nodes $A \subseteq V$, we thus search for a partition $S \setminus A = S_1 \cup S_2 \cup S_3$ with $|S_1|, |S_2| \leq (\alpha + \epsilon)k$ and $|S_3| \leq (1 - 2\alpha + 2\epsilon)k$. For α close to $\frac{1}{2}$, this takes time of order

$$O^*(\binom{k}{(1-2\alpha-2\epsilon)k} 2^{(2\alpha+2\epsilon)k})$$

which also gives the total time bound for the second phase of the algorithm.

Setting $\epsilon = 0$, we obtain an upper bound on the total computation time by solving

$$\binom{k}{\alpha k} 2^{\alpha k} = \binom{k}{(1-2\alpha)k} 2^{2\alpha k}$$

or, according to Stirling's Formula

$$(\frac{1}{\alpha})^{\alpha}(\frac{1}{1-\alpha})^{1-\alpha} = (\frac{1}{2\alpha})^{2\alpha}(\frac{1}{1-2\alpha})^{1-2\alpha} 2^{\alpha}.$$

The solution of this equation is $\alpha < 0.436$. Hence we can achieve a total time bound of

$$[(\frac{1}{0.436})^{0.436}(\frac{1}{0.564})^{0.564} 2^{0.436}]^k = 2.684^k$$

by an appropriately small choice of $\epsilon > 0$.

4 The rectilinear case

Given a set $S = \{s_1, \ldots, s_k\}$ of points in the plane, the *rectilinear* Steiner tree problem asks for a shortest tree connecting the points in S, relative to the so-called Manhattan-metric (where the distance between two points is, by

definition, the sum of the differences of their x- and y-coordinates). Equivalently, we may define an instance of the Steiner tree problem *rectilinear*, if the underlying graph $G = (V, E)$ is a grid graph (the so-called "Hannan-grid") in the plane (with the grid being generated by the x- resp. y-coordinates of the points in S). We refer the reader to [8] for an introduction to the rectilinear case.

According to Ganley and Cohoon [4] and Fößmeier and Kaufmann [3], the dynamic programming approach for computing minimum Steiner trees can be implemented more efficiently in the rectilinear case as follows. The basic notion is that of a full Steiner tree: If $X \subseteq S$ is given, a minimum Steiner tree $T = T(X)$ for X is *full* if each node in X is a leaf of T. We call $X \subseteq S$ a *full set* if every minimum Steiner tree for X is full.

Clearly, every minimum Steiner tree $T = T(X)$ for $X \subseteq S$ decomposes uniquely into *full components*, *i.e.*, edge-disjoint full subtrees. A crucial result of Hwang [6] states that, in the rectilinear case, full components (sets) can be assumed to have a certain simple topological structure. Subsets $X \subseteq S$ with this particular structure are called *candidate full sets*. The set of all candidate full sets $X \subseteq S$ is denoted by $\mathcal{F}(S)$. Given a candidate full set $X \in \mathcal{F}(S)$, one can (due to the particular simple structure of full components) easily compute (in linear time) a corresponding *candidate full tree* $T_{full}(X)$, which is guaranteed to be a minimum Steiner tree for X in case X is a full set.
Adopting the notation

$$X = X_1 \bowtie X_2 \;\; \Leftrightarrow \;\; X = X_1 \cup X_2, |X_1 \cap X_2| = 1$$

from [4], we may thus compute minimum Steiner trees for all $X \subseteq S$ by means of the recursion

$$T(X) \;\; = \;\; \min \; T_{full}(X_1) \; \cup \; T(X_2), \tag{3}$$

where the minimum is taken over all decompositions $X = X_1 \bowtie X_2$ with $X_1 \in \mathcal{F}(S)$ and $|X_1| \geq 2$. Note that when $X \subseteq S$ itself is a full set, then $X \in \mathcal{F}(S)$, so we may take $X_1 = X$ and let X_2 be a singleton.

The running time of this procedure depends on the number of candidate full sets. Indeed, letting

$$\mathcal{F}(X) := \{X_1 \in \mathcal{F}(S) \mid X_1 \subseteq X\},$$

we find that computing the minimum in (3) takes time $O^*(|\mathcal{F}(X)|)$ – assuming recursively that $T(X_2)$ is known already for all subsets $X_2 \in S$ of size

$|X_2| < |X|$. (Recall that, as mentioned above, $T_{full}(X_1)$ can be computed for given $X_1 \in \mathcal{F}(S)$ in time $O(|X_1|) = O(k) = O^*(1)$.)

The main result of Ganley and Cohoon [4] states that (due to the specific topological structure of full sets), only very few subsets of X are candidate full sets. More precisely, they show that for $|X| = i$ we have $|\mathcal{F}(X)| \leq 1.62^i$. This bound is further improved by Fößmeier and Kaufmann [3] to $|\mathcal{F}(X)| \leq 1.386^i$. As a consequence, the total running time of the recursion, applying (3) to all sets $X \subseteq S$ with increasing size $|X| = i$, can be bounded by

$$O^*(\sum_{i=1}^{k} \binom{k}{i} 1.386^i) \quad = \quad O^*(2.386^k). \tag{4}$$

Applying our splitting technique to this recursion, we would—just like in section 3—compute the minimum Steiner trees only up to a certain level $i = \alpha k$, $\alpha < \frac{1}{2}$. The time consumed by this computation is

$$O^*(\sum_{i=1}^{\alpha k} \binom{k}{i} 1.386^i) \quad = \quad O^*(\binom{k}{\alpha k} 1.386^{\alpha k}). \tag{5}$$

On the other hand, searching for the unknown 3-split would roughly (we set $\epsilon = 0$) take

$$O^*(\binom{k}{(1-2\alpha)k} 2^{2\alpha k}). \tag{6}$$

Again, the best upper bound on the running time of our algorithm is obtained by balancing (5) and (6). For $\alpha \approx 0.477$, we obtain an upper bound of $O^*(2.335^k)$ – a minor improvement over the original bound (4).

There is one problem that we are left to solve: Recall that in "phase 1" of our algorithm we compute small Steiner trees up to level $i = \alpha k$ not only in G, but also in certain contracted graphs. But these graphs are in general not rectilinear anymore! A moment's thought, however, reveals that this problem as simply non-existent. Indeed, the only reason for considering contracted graphs in section 3 is notational convenience: Assume, for example, that we are to compute (recursively) the minimum Steiner tree for a certain subset

$$X \cup v_A \quad \text{in } G/A, \quad X \subseteq V \setminus A.$$

This is tantamount to looking for a *minimum Steiner A-forest* for X in G, *i.e.*, a minimum length forest $F \subseteq E$, connecting all of X to A. In other

words, a Steiner A-forest for X consists of $|A|$ tree components ($|A| \geq 1$), each containing exactly one node of A. Thus a minimum Steiner A-forest $\mathcal{F} \subseteq E$ gives rise to a minimum Steiner tree F in G/A and conversely.

Rather than computing minimum Steiner trees for various sets $X \cup v_A$ in certain contracted graphs G/A, we compute minimum Steiner A-forests in G for various sets X and A. In the rectilinear case, this can be done in complete analogy to the full set dynamic programming approach described above.

For $X \subseteq S \setminus A$, $|A| \geq 1$, let $F_A(X \cup A)$ denote the minimum Steiner A-forest for X. Then $F_A(X \cup A)$ consists of at most $|A|$ nonempty tree components. (Recall that we always consider a tree as a set of edges. So a tree component consisting of a single vertex $a \in A$ is empty.) Each such nonempty tree component contains exactly one node $a \in A$ and decomposes into one or more full components. Thus we can compute $F_A(X \cup A)$ recursively from $F_A(A) = \varnothing$ and

$$F_A(X \cup A) \;=\; \min\; T_{full}(X_1) \;\cup\; F_A(X_2 \cup A) \tag{7}$$

where the minimum is taken over all candidate full sets $X_1 \in \mathcal{F}(X \cup A)$ with $X \cup A = X_1 \bowtie (X_2 \cup A)$.

Note that the dynamic program (7) is (for fixed A) very similar to (3). (Indeed, we formally obtain (4) from (7) by setting $A = \varnothing$.) This completes our proof of the upper bound on the running time.

5 Concluding remarks

We presented a splitting technique to speed up the dynamic programming approach to minimum Steiner tree computation. We do not claim that our improvements as presented in sections 3 and 4 are of any practical use. Yet it might turn out that already for small values of $|A|$, say $|A| < 4$, the existence of 2-splits with $|S_i|$ fairly close to $k/2$ can be guaranteed. This needs to be further investigated, as it might well be of practical interest.

References

[1] C. Gröpl, S. Hougardy, T. Nierhoff and H. J. Prömel, (2001) Approximation Algorithms for the Steiner Tree Problem in Graphs, in: *Steiner Trees in Industries*, X. Cheng and D.-Z. Du (eds.), Kluwer.

[2] S. E. Dreyfus and R. A. Wagner, (1972) The Steiner problem in graphs. *Networks*, 1, 195–207.

[3] U. Fößmeier and M. Kaufmann, (2000) On exact solutions for the rectilinear Steiner tree problem Part 1: Theoretical results. *Algorithmica*, 26, 68–99.

[4] J. L. Ganley and J. P. Cohoon, (1994) Optimal rectilinear Steiner minimal trees in $O(n^2 2.62^n)$ time, in: *Proceedings of the Sixth Canadian Conference on Computational Geometry*, 308–313.

[5] M. Garey and D. Johnson, (1977) The Rectilinear Steiner Tree Problem is NP-Complete, *SIAM Journal on Applied Mathematics* 32(4), 826–834.

[6] F. K. Hwang, (1976) On Steiner Minimal Trees with Rectilinear Distance. *SIAM Journal on Applied Mathematics*, 30(1), 104–114.

[7] D. M. Warme, P. Winter and M. Zachariasen, (2000) Exact Algorithms for Plane Steiner Tree Problems: A Computational Study, in: *Advances in Steiner Trees*, D.Z. Du, J.M. Smith and J.H. Rubinstein (eds.), Kluwer.

[8] M. Zachariasen, The Rectilinear Steiner Tree Problem: A Tutorial, in: *Steiner Trees in Industries*, X. Cheng and D.-Z. Du (eds.), Kluwer.