A Faster Dynamic Programming Algorithm for Exact Rectilinear Steiner Minimal Trees

Joseph L. Ganley and James P. Cohoon*
Department of Computer Science
University of Virginia
Charlottesville, Virginia 22903

Abstract

An exact rectilinear Steiner minimal tree algorithm is presented that improves upon the time and space complexity of previous guarantees and is easy to implement. Experimental evidence is presented that demonstrates that the algorithm also works well in practice.

1 Introduction

The *rectilinear Steiner minimal tree (RSMT)* problem is stated as follows: given a set of points called *terminals* in the plane, find a set of horizontal and vertical line segments of minimal total length that interconnects the terminals. The RSMT problem is similar to the well-known minimum spanning tree problem, with one important exception: in a minimum spanning tree, terminals may only be connected to one another, while in the RSMT problem, new points called *Steiner points* may be introduced such that the length of the tree is reduced. Figure 1 illustrates an optimal RSMT for a set of 20 terminals. Garey and Johnson [3] prove that the RSMT problem is NP-complete, indicating that a polynomial-time algorithm to solve it exactly is unlikely to exist. However, a primary application of RSMT algorithms is routing of VLSI circuits. In such applications, the number of terminals is typically quite small, so an efficient exact algorithm may often be practically applied. In this paper we present a dynamic programming algorithm that computes an exact RSMT and has time and space complexity better than any previous worst-case guarantees. The algorithm is easy to implement, and we present empirical evidence that it performs well in practice on sets of terminals small enough to solve practically.

*The authors gratefully acknowledge the support of National Science Foundation grant MIP-9107717 and Virginia CIT award 5-30971.

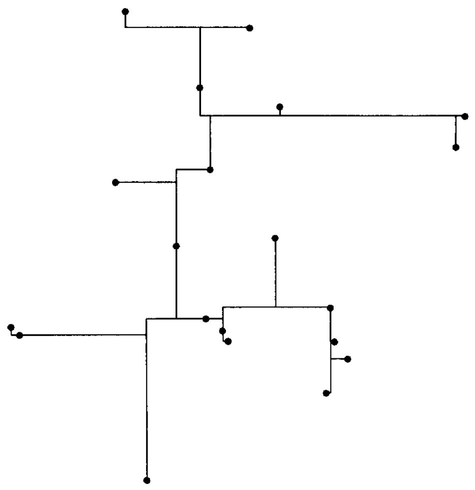

Figure 1: An optimal RSMT for a set of 20 terminals.

2 Previous work

Previous work on exact RSMT algorithms can be categorized into two types, based on whether they attack the RSMT problem directly or reduce it to the Steiner problem in graphs.

There have been few algorithms that use geometric approaches to solve the RSMT problem directly. Yang and Wing [14] present a branch-and-bound algorithm with worst-case time complexity $O(2^{k^2})$, where k is the number of terminals. The largest instance on which they test their algorithm contains 9 terminals. Wong and Pecht [13] describe another exact RSMT algorithm that is essentially an exhaustive version of the edge-embedding heuristic of Ho, Vijayan, and Wong [5]. They do not state its time complexity, but it appears to be at least exponential in the number of edges in the Hanan grid graph (see next paragraph), of which there are $O(k^2)$. Their algorithm is applicable

1066-1395/94 $3.00 © 1994 IEEE

to instances containing up to about 15 terminals [12]. Thomborson, Deneen, and Shute [11] describe an algorithm whose time complexity is $O(2^{\sqrt{k}\log k})$, but their algorithm is designed such that either the time complexity or the optimality of the computed tree is probabilistic. Furthermore, while their algorithm has not been implemented, it is not expected to be competitive for small terminal sets [10]. Salowe and Warme [9] present an algorithm that works very well in practice—it can efficiently solve problems with up to 30 terminals—but the only known bound on its time complexity is $O(2^{2^k})$.

The other major approach to solving the RSMT problem exactly is to reduce it to the Steiner problem in graphs. Hanan [4] proves that a rectilinear Steiner minimal tree exists for every set of terminals, that is composed solely of subsegments of the set of horizontal and vertical lines (called *grid lines*) that contain terminals. Thus, one can construct a *grid graph* whose vertices are the terminals and the points where the grid lines intersect (the latter are potential Steiner points), and in which there is an edge between pairs of vertices that are adjacent along a grid line. The weight of each edge is the rectilinear distance between its endpoints. It is clear from Hanan's theorem that an optimal solution to the Steiner problem in the grid graph is an optimal solution to the original geometric problem. Thus, a common approach to solving the RSMT problem is to apply an algorithm for the Steiner problem in graphs to the Hanan graph. Currently the algorithm with the best guaranteed time complexity for this task is the dynamic programming algorithm of Dreyfus and Wagner [2], which has time complexity $O(k^2 3^k + (k^2 \log k)2^k)$ in planar graphs [1] such as the Hanan grid graph. Since the Dreyfus-Wagner algorithm has the best worst-case time complexity for solving the RSMT problem exactly, and since it is somewhat similar to our algorithm, it is against the Dreyfus-Wagner algorithm that we compare ours.

3 The algorithm

Before describing the algorithm we must define a few terms. A set T of terminals is a *full set* if, in every optimal RSMT for T, every terminal in T is a leaf. An RSMT of a full set is called a *full tree*. Hwang [6] proved that a full tree can have only one of two simple topologies. The first type of topology is a *backbone* segment adjacent to one of the extreme terminals, with alternating segments connecting the

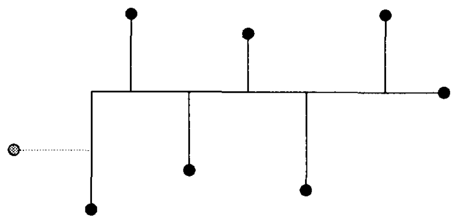
Figure 2: Possible full tree topologies according to Hwang's theorem (the shaded segment and terminal may or may not be present).

other terminals to the backbone. The second type is the same as the first, but with one additional terminal connected to the segment that connects the other extreme terminal to the backbone. These two topologies are illustrated in Figure 2. Using Hwang's theorem, an optimal RSMT of a full set can be computed in linear time by checking these topologies. Finally, a well-known theorem regarding RSMTs is that every RSMT consists of a number of full trees that intersect at terminals of degree two or greater (see, e.g., Hwang, Richards, and Winter [7]).

From these facts it is clear that for any set of terminals, an optimal RSMT is either a full tree satisfying Hwang's theorem, or can be divided into two smaller trees joined at a terminal. This observation leads directly to the dynamic programming algorithm. Subsets of the input set of terminals are enumerated; for pairs of terminals, the optimal RSMT is simply an edge between the terminals. For subsets of more than two terminals, the algorithm checks the length of the tree produced by applying Hwang's theorem and the lengths of the trees produced by joining the optimal RSMTs of every pair of disjoint subsets at every terminal, and chooses the decomposition with minimum length. The subsets are enumerated in order of cardinality, so that at each step, the optimal RSMT for every smaller subset has already been computed and stored. Figure 3 describes the algorithm in detail. The algorithm shown computes only the length of the optimal tree; a similar second pass computes the actual tree given the lengths computed by the first pass. We henceforth refer to this algorithm as the *Full set Dynamic Programming (FDP)* algorithm.

4 Analysis

The FDP algorithm described above improves upon the Dreyfus-Wagner algorithm with respect to both

```
FDP(set of terminals $T$)
(1) For $m = 2$ to $|T|$
(2)     For each $C \subseteq T$ such that $|C| = m$
(3)         $\ell[C] =$ FullTree($C$)
(4)         For each $i \in C$
(5)             For each $F \subset C - \{i\}$
(6)                 $\ell' = \ell[f \cup \{i\}] + \ell[C - F]$
(7)                 $\ell[C] = \min\{\ell[C], \ell'\}$
```

Figure 3: The FDP algorithm. The routine **Full-Tree** computes the length of an optimal full tree using Hwang's theorem.

time and space complexity. The Dreyfus-Wagner algorithm has time complexity $O(n3^k + (n \log n)2^k)$ in planar graphs [1], where k is the number of terminals and n is the number of candidate Steiner points. In the Hanan grid graph, n is $O(k^2)$, making the overall time complexity $O(k^2 3^k + (k^2 \log k)2^k)$. The space complexity of the Dreyfus-Wagner algorithm is $O(n2^k)$, i.e., $O(k^2 2^k)$.

The time complexity of the FDP algorithm is $O(k3^k + k2^k)$, and its space complexity is $O(2^k)$. The time complexity is derived in a manner similar to that of Dreyfus and Wagner [2]. Recall Figure 3: for each value of m in loop (1),

- ◇ Loop (2) iterates $L_2 = \binom{k}{m}$ times.
- ◇ Loop (4) iterates $L_4 = mL_2$ times.
- ◇ Loop (5) iterates $2^{m-1}L_4$ times.

Thus, the time complexity of the FDP algorithm is

$$\sum_{m=2}^{k} \binom{k}{m} m2^{m-1} = k3^{k-1}$$

This analysis does not include line (3) of Figure 3, in which a full tree is computed for every subset of T, each in linear time, resulting in an additional $O(k2^k)$ time. Thus, the total time complexity of the FDP algorithm is $O(k3^k + k2^k)$[†].

The space complexity is straightforward: the algorithm stores only the length of the optimal tree for each subset; there are 2^k subsets, and thus the space complexity is $O(2^k)$. In practice, one might wish to store the optimal decomposition of each subset along with its length. This does not change the time complexity of the algorithm, but eliminates the need for the second pass to compute the actual RSMT, and thus substantially reduces the running time in practice. This change increases the space requirement to $O(k2^k)$, which is still an improvement over the Dreyfus-Wagner algorithm.

[†]This is, of course, asymptotically equivalent to $O(k3^k)$, but we retain the other term for consistency with Dreyfus and Wagner [2] and because for small terminal sets, the second term is significant.

$\|T\|$	FDP	DW	$\|T\|$	FDP	DW
5	0.004	0.091	13	15.21	351.5
6	0.010	0.169	14	49.97	1274
7	0.027	0.394	15	176.7	4380
8	0.061	1.093	16	598.2	35141
9	0.174	3.248	17	2008	—
10	0.493	10.05	18	6615	—
11	1.532	31.00	19	22104	—
12	4.773	98.69	20	72574	—

Table 1: Average running times (in seconds) for the FDP algorithm vs. the Dreyfus-Wagner (DW) algorithm.

5 Empirical results

We have implemented the FDP algorithm and the Dreyfus-Wagner algorithm to demonstrate that the FDP algorithm is faster not only in theory, but also for sets of terminals small enough to be solved on a workstation in under 24 hours. Both algorithms are implemented in C and run on a Sun Sparc-2™ workstation. The Dreyfus-Wagner algorithm is applied to the Hanan grid graph with all vertices outside the rectilinear convex hull of the terminals deleted; Yang and Wing [14] show that this reduction does not affect the optimality of the resulting RSMT.

Table 1 shows the average running time for each algorithm on sets of terminals of size 5 to 20, generated uniformly at random over a 10,000 by 10,000 grid. Each value is averaged over ten sets of terminals; the running time of both algorithms is very consistent, and their variances are quite low. Not only does the FDP algorithm improve upon the Dreyfus-Wagner algorithm asymptotically, but these experiments demonstrate that it is also a dramatic improvement for sets of terminals small enough to solve practically. The missing entries for the Dreyfus-Wagner algorithm could not be run on our workstation due to insufficient memory. This illustrates the fact that in practice, the real difficulty with the Dreyfus-Wagner algorithm is often not its time complexity but its space requirements. The drastic increase in running time for the Dreyfus-Wagner algorithm between 15 and 16 terminals is caused by thrashing due to these large space requirements. Note that this implementation of the Dreyfus-Wagner algorithm can solve 16-terminal problems in under 24 hours on our workstation, while the FDP algorithm solves 20-terminal problems within the same period of time.

It should be noted that in VLSI routing applications, the number of terminals is typically quite small—for example, in the benchmark instance PRIMARY 1 [8], the largest net contains 18 terminals, and the average net contains far fewer. Furthermore, the FDP algorithm is easy to implement—the implementation tested here was written in only a few hours.

6 Conclusions

We have presented a dynamic programming algorithm called *Full set Dynamic Programming (FDP)* that computes exact rectilinear Steiner minimal trees and has worst-case time and space complexity better than any previous guarantees. In particular, we improve upon the best previous guarantee, the dynamic programming algorithm of Dreyfus and Wagner [2]. While the Dreyfus-Wagner algorithm requires $O(k^2 3^k + (k^2 \log k) 2^k)$ time and $O(k^2 2^k)$ space, the FDP algorithm requires $O(k3^k + k2^k)$ time and $O(2^k)$ space. Furthermore, the FDP algorithm is far faster in practice than the Dreyfus-Wagner algorithm, and is very easy to implement.

Thomborson, Alpern, and Carter [10] present several optimizations to the Dreyfus-Wagner algorithm that reduce the constant factors in its time complexity. These optimizations rely on explicit memory management and loop reordering to maximize locality of reference, and on some geometric properties of the Hanan grid graph. Many of these optimizations are applicable to the FDP algorithm as well.

Our work continues toward an exact RSMT algorithm with time complexity $O(ka^k + k2^k)$ for some $a < 3$. We believe this might be accomplished by eliminating some subsets from consideration in the innermost loop of Figure 3.

References

[1] M. BERN, *Faster exact algorithms for Steiner trees in planar networks*, Networks, 20 (1990), pp. 109–120.

[2] S. E. DREYFUS AND R. A. WAGNER, *The Steiner problem in graphs*, Networks, 1 (1972), pp. 195–207.

[3] M. R. GAREY AND D. S. JOHNSON, *The rectilinear Steiner tree problem is NP-complete*, SIAM Journal of Applied Mathematics, 32 (1977), pp. 826–834.

[4] M. HANAN, *On Steiner's problem with rectilinear distance*, SIAM Journal of Applied Mathematics. 14 (1966), pp. 255–265.

[5] J. HO, G. VIJAYAN, AND C. K. WONG, *New algorithms for the rectilinear Steiner tree problem*. IEEE Transactions on Computer-Aided Design, 9 (1990), pp. 185–193.

[6] F. K. HWANG, *On Steiner minimal trees with rectilinear distance*, SIAM Journal of Applied Mathematics, 30 (1976), pp. 104–114.

[7] F. K. HWANG, D. S. RICHARDS, AND P. WINTER, *The Steiner Tree Problem*, North-Holland, Amsterdam, Netherlands, 1992. (Annals of Discrete Mathematics 53).

[8] B. T. PREAS, *Benchmarks for cell-based layout systems*, in Proceedings of the Twenty-fourth Design Automation Conference, 1987, pp. 319–320.

[9] J. S. SALOWE AND D. M. WARME, *An exact rectilinear Steiner tree algorithm*, in Proceedings of the International Conference on Computer Design, 1993, pp. 472–475.

[10] C. D. THOMBORSON, B. ALPERN, AND L. CARTER, *Rectilinear Steiner tree minimization on a workstation*, in Proceedings of the DIMACS Workshop on Computational Support for Discrete Mathematics, 1992.

[11] C. D. THOMBORSON, L. L. DENEEN, AND G. M. SHUTE, *Computing a rectilinear Steiner minimal tree in $O(n^{O(\sqrt{n})})$ time*, in Parallel Algorithms and Architectures, A. Albrecht, ed., Springer-Verlag, Berlin, Germany, 1987, pp. 176–183.

[12] Y. T. WONG, *personal communication*. 1993.

[13] Y. T. WONG AND M. PECHT, *A solution for Steiner's problem*, in Placement and Routing of Electronic Modules, M. Pecht, ed., Marcel Dekker, New York, NY, 1993, pp. 261–304.

[14] Y. Y. YANG AND O. WING, *Optimal and suboptimal solution algorithms for the wiring problem*, in Proceedings of the IEEE International Symposium on Circuit Theory, 1972, pp. 154–158.