Submitted to IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 1993.

Optimal Rectilinear Steiner Tree Routing in the Presence of Obstacles

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

***Abstract*—This paper presents a new model for VLSI routing in the presence of obstacles, that transforms any routing instance from a geometric problem into a graph problem. It is the first model that allows computation of *optimal* obstacle-avoiding rectilinear Steiner trees in time corresponding to the instance size (the number of terminals and obstacle border segments) rather than the size of the routing area. For the most common multi-terminal critical nets—those with three or four terminals—we observe that optimal trees can be computed as efficiently as good heuristic trees, and present algorithms that do so. For nets with five or more terminals, we present algorithms that heuristically compute obstacle-avoiding Steiner trees. Analysis and experimental results demonstrate that the model and algorithms work well in both theory and practice. Also presented are several theoretical results: a derivation of the Steiner ratio for obstacle-avoiding rectilinear Steiner trees, and complexity results for two special cases of the problem.**

1 Introduction

In VLSI design automation, a fundamental task is routing a net. Typically, this routing is performed in the presence of obstacles that the wires of the net must not intersect, such as logic cells and wires in previously routed nets. This problem has been well-studied for the case of two-terminal nets. Initially, Lee [17] and Moore [21] independently devised the technique of *maze routing* for optimally routing two-terminal nets in what were then reasonably-sized problem instances. The major drawback of maze routing is that its time and space demands are a function of the size of the routing area, rather than the size of the actual problem instance (the number of terminals and obstacle border segments). At the time, these demands were reasonable, but presently the routing area of typical VLSI instances is quite large. Later, Hightower [15] devised a technique called *line search routing (LSR)* with much lower space demands, but LSR is a heuristic, i.e. it produces suboptimal solutions. Later still, Cohoon and Richards [5] devised the first algorithm that optimally routes two-terminal nets in the presence of obstacles, in time corresponding to the size of the routing instance rather than to the routing area. Lee, Yang, and Wong [18] survey algorithms for two-terminal routing in the presence of obstacles.

The problem of routing a multi-terminal net in the presence of obstacles has received substantially less attention. As a result, a VLSI designer is forced to use a multi-terminal variant of a maze routing algorithm, which incurs the same space demands as the two-terminal variety, and often compute solutions that are far from optimal. Other variants can produce optimal routing solutions, but still require time and space corresponding to the size of the routing area.

Even in the absence of obstacles, multi-terminal routing corresponds to the *rectilinear Steiner tree (RST)* problem, which is NP-complete [10]. This suggests that no polynomial-time algorithm can solve the RST problem exactly. Nonetheless, exponential-time algorithms have been devised that can solve the RST problem exactly for small instances in reasonable time [9, 23, 25], as have many efficient polynomial-time heuristics (see Hwang, Richards, and Winter [16]) that produce good suboptimal solutions. The time complexity of the vast majority of these algorithms is a function of size of the instance, not of the routing area as in maze routing algorithms. No such algorithms have existed previously for computing RSTs in the presence of obstacles, a problem that we refer to as the *obstacle-avoiding rectilinear Steiner tree (OARST)* problem. Here, we present a theorem, analogous to Hanan's theorem [13] for the standard RST

Figure 1: The escape graph (see text).

a generalization of the line search escape segments used by Hightower's line search routing heuristic [15].

To describe the escape graph, we appeal to an analogy to interstate highway travel. An obstacle corresponds to a city. On every side of an obstacle, we generate an escape segment that forms a portion of the obstacle's 'beltway.' For the instance depicted in Figure 1(a), these are the dashed segments of Figure 1(b). To enable connections between obstacles, each beltway escape segment is extended to also form a 'highway' escape segment. A highway escape segment is a maximal segment with respect to the routing region, i.e. it ends with its abutment to either an obstacle border segment or the internal perimeter of the routing region. Finally, we introduce segments that extend from the terminals in all unobstructed directions. These segments are also maximal in a manner similar to the highway escape segments. The dashed segments shown in Figure 1(c) are the escape segments for the instance depicted in Figure 1(a).

It has been shown that escape segments suffice for optimal routing of two-terminal nets [5]. We now proceed to show that there is an optimal routing for any net with $k > 2$ terminals, that uses only escape segments[1].

Theorem 1 *If an instance of the OARST problem is solvable, then there is an optimal solution composed only of escape segments.*

Proof: Suppose there exists a k-terminal problem instance I, with $k > 2$, such that all optimal Steiner trees for I contain at least one segment that is not an escape segment. We will show that this supposition leads to a contradiction.

Let τ be an optimal Steiner tree for I that contains a minimal number of non-escape segments among optimal Steiner trees for I. Let segment s be a routing segment in τ that is not an escape segment. Without loss of generality, assume that s is horizontal.

Obviously, s has two endpoints a and b beyond which no further colinear segment is incident. There may be segments incident and orthogonal to s. In fact, there must be orthogonal segments incident to a and b. If either a or b did not have an orthogonal segment incident to it, then it would be a terminal, contradicting the assumption that s is not an escape segment.

Let u be the number of orthogonal segments incident to s from above, and let d similarly be the number of orthogonal segments incident from below. Colinear segments that are incident to s both from above and below are considered two distinct segments separated by s.

If u is equal to d, then slide s up until it is colinear with some escape segment. We know there is room to slide, as s is not an escape segment. An escape segment above s must exist, since the routing region perimeter is itself inscribed by escape segments. Since the length of the segments above s decreases by exactly the amount that the length of the segments below s increases, the tree resulting from this maneuver has the same length as τ; hence, it is optimal. In addition, any vertical segment incident to s that was an escape segment remains an escape segment. In fact, all segments that were escape segments before the slide remain so. Thus, the tree resulting from this sliding maneuver contradicts our assumption that τ contains a minimal number of non-escape segments.

If instead, u is greater than (less than) d, then we may slide s up (down), decreasing the length of the tree and contradicting its optimality. We again know there is room to slide, since s is not an escape segment.

This completes the case analysis. We have shown that every solvable instance of the OARST problem has an optimal solution composed only of escape segments. □

Theorem 1 can be used similarly to Hanan's well-known theorem [13] for the standard RST problem, to construct a graph representation of a routing instance from its geometric description. The vertices of the graph are the terminals and the points at which the escape segments intersect (which are potential Steiner points). An edge exists between two vertices if they lie on the same escape segment, and if no other vertex lies between them on that same escape segment. The weight of an edge is simply the rectilinear distance between its endpoints. We call this graph the *escape graph*. It is obvious from Theorem 1 that an optimal solution to the Steiner problem on the escape graph for a particular routing instance is an optimal solution to the original instance. Thus, given an escape graph, the routing problem from which it was generated is solved by finding a solution to the Steiner problem in the escape graph.

3 Escape Graph Generation

The generation of the escape segments from a problem instance is relatively straightforward and is performed by adapting standard computational geometry

[1] Throughout the paper, we use the term *escape segment* interchangeably to mean either a complete escape segment or a subsegment of a complete escape segment.

techniques. First, horizontal and vertical line sweeps of the boundary segments of the obstacles are performed to construct the escape segments [5]. The sweeps are done in $O(m \log m)$ time, where m is the number of obstacle boundary segments.

The horizontal sweep at a given x-coordinate is achieved by performing a binary search to determine the endpoints of the potential escape segment associated with the current boundary segment. There is a potential escape segment above the boundary segment if the boundary segment represents an upper contour and similarly below the boundary segment if it represents a lower contour. For a horizontal boundary segment that corresponds to a previously placed wire, there can be escape segments both above and below the boundary segment.

The final determination of whether a potential escape segment exists requires examining the associated area of the layout surface to see whether it is a routing region. This determination is done in constant time per escape segment by examining the line sweep information for the previous and successor line sweep locations [5].

The processing of a vertical sweep is performed in an analogous manner.

The intersections of the escape segments are then computed. Since the escape segments are generated in sorted order, the intersections of the escape segments are determined in $O(m + n)$ time, where n is the number of intersections [2]. If there are m escape segments, then n is $O(m^2)$ in the worst case (if all the horizontal escape segments intersect all the vertical escape segments). Construction of the escape graph from the escape segments and their intersections can be accomplished in $O(n)$ time in a straightforward manner; this procedure is not discussed further. The total time complexity of generating the escape graph is thus $O(\max\{n, m \log m\})$, where n is $O(m^2)$ in the worst case.

4 Escape Graph Reduction

Often, many of the vertices in the escape graph can be deleted, along with their adjacent edges, while still guaranteeing that an optimal solution exists that is constrained to the escape graph. We now describe several tests, the application of which eliminates many vertices from the escape graph to produce a *reduced escape graph*.

The first test is called the *shortest-path test*. The idea is that any vertex that is not a terminal, and that appears in an optimal OARST, must lie on a shortest path between some pair of terminals. This fact is fairly obvious; if an optimal tree contains a vertex v that does not satisfy this criterion, then v can be replaced by another vertex that is on a shortest path between the terminals to which v is connected, reducing the length of the tree and contradicting its optimality. Thus, we simply examine each vertex in the escape graph, and eliminate those that do not lie on a shortest path between some pair of terminals. This involves examining every vertex in the escape graph, with respect to every pair of terminals. If an all-pairs shortest paths matrix is available, this procedure incurs a time complexity of $O(k^2 n)$, where k is the number of terminals and n is the number of potential Steiner points. Since the escape graph is planar, all-pairs shortest paths information can be computed in $O(n^2)$ time [8]; this information will also be used later in the computation of Steiner trees.

The second test is the *dimension reduction test* described by Yang and Wing [26] for the standard RST problem. Yang and Wing prove that if a vertex v is a corner vertex, i.e. it is incident to exactly two orthogonal edges e_1 and e_2, and v is not a terminal, and edges exist that form the other two sides of a rectangle with e_1 and e_2, then v, e_1, and e_2 can be deleted. The proof of this theorem holds for escape graphs as well: any path that might pass through v can be replaced by a path with equal length that instead passes through the sides of the rectangle opposite e_1 and e_2. This test has $O(n)$ time complexity.

Finally, it is clear that any nonterminal vertex v of degree 2 that remains after the dimension reduction test can be eliminated, and its neighbors connected directly by an edge whose weight is the sum of the weights of the edges adjacent to v, and that any nonterminal of degree 1 can be deleted along with its adjacent edge.

Figure 2 shows the escape graph and reduced escape graph for a randomly generated OARST instance with 10 terminals and 10 rectangular obstacles.

Empirically, the average fraction of the vertices that are eliminated by the above tests on randomly generated instances is dependent only on k, i.e. is independent of the number n of vertices. Figure 3(a) shows the percentage of vertices that are eliminated as a function of k. Figure 3(b) shows the average number of vertices n *before reduction* as a function of the number m of obstacle boundary segments. For a given k and m, one can expect the number of vertices in the reduced escape graph to be, on average, the number of Steiner points indicated by Figure 3(b) for the given m

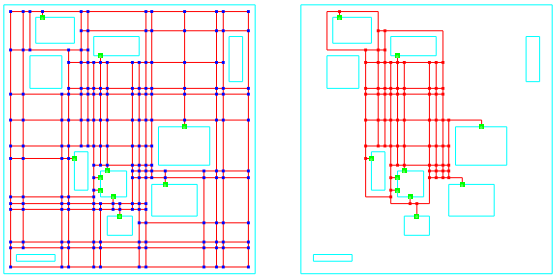
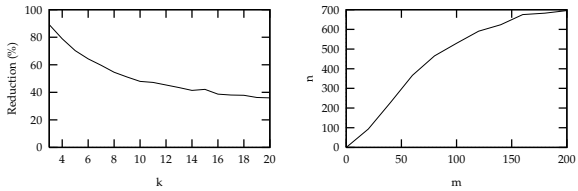


Figure 2: Escape graph reduction.



(a) (b)

Figure 3: (a) Percentage of vertices eliminated by reduction tests as a function of number of terminals k and (b) Number of vertices n in the *unreduced* escape graph, as a function of number of obstacle border segments m.

reduced by the percentage indicated by Figure 3(a) for the given k.

Note that most of the statistics in this paper are gathered from randomly generated instances. It might be argued that random instances are not representative of problems that arise in practice, but they do form a worst case of sorts, in that more structure in the layout of obstacles results in redundant escape segments and fewer vertices in the escape graph. In particular, for structured design methodologies such as standard cell, the escape graph contains $O(m)$ vertices. Note also the shape of the curve in Figure 3(b); the decrease in the slope of the curve as m grows is due to increasing density, and thus more redundant escape segments.

5 Exact Algorithms

Typically, exact solutions to NP-complete problems are infeasible in practice. However, it is often the case that small instances can be solved practically. For the OARST problem, the escape graph model enables us to compute optimal Steiner trees for three- or four-terminal nets as efficiently as a typical heuristic solution.

Terminals	2	3	4	5	6
Percentage	52.8	27.5	7.4	2.7	3.1
Terminals	7	8	9	10	>10
Percentage	1.4	0.2	1.0	0.1	3.6

Table 1: Distribution of terminals per net in Primary 1.

A folk theorem of VLSI routing is that most nets contain four or fewer terminals. In an effort to verify this claim, we examine the SIGDA Benchmark Suite [22]. Table 1 shows the distribution of terminals per net in the benchmark instance Primary 1. As can be seen in the table, three- and four-terminal nets comprise just over one-third of the total nets. The largest net contains 18 terminals, and the average number of terminals per net is 3.25. The distribution for other benchmarks is similar.

For a three-terminal net, an optimal Steiner tree can have only one of two topologies. It can be a simple path between the terminals, or all three terminals can be connected to a single Steiner point. Thus, an optimal OARST for a three-terminal net may be computed as follows. The length of each of the three possible simple paths is checked, as well as the length of every tree formed by connecting the three terminals to each candidate Steiner point, and the tree with minimum length is optimal. The latter topology—where the tree contains a Steiner point—dominates the computation time, and is examined in $O(n)$ time, where n is the number of candidate Steiner points, assuming all-pairs shortest paths information is available. Since the escape graph is planar, all-pairs shortest paths can be computed in $O(n^2)$ time by the algorithm of Frederickson [8].

Similar observations can be made for four-terminal nets. For four terminals, the following topologies are possible:

- A simple path through the four terminals.
- A *star* in which three terminals are all connected to the fourth.
- A *cross* in which all four terminals are connected to a single Steiner point.
- A T in which three terminals are connected to a single Steiner point, and the fourth terminal is connected to one of those three terminals.
- An H in which two terminals are connected to each of two Steiner points, which are connected to each other.

These topologies are illustrated in Figure 4. There are twelve possible orderings of the simple path topol-

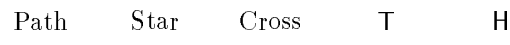


Figure 4: The possible topologies for four terminals.

$k = 3$		$k = 4$	
EE	STE	EE	STE
0.04	1.45	0.08	9.06

Table 2: Average running times (in seconds) of explicit enumeration (EE) and spanning tree enumeration (STE) algorithms.

ogy, four for the star, one for the cross, twelve for the T, and six for the H, totaling 35 distinct topological instances—a sufficiently small number to examine explicitly. An optimal Steiner tree for a four-terminal net can be efficiently computed by enumerating these topologies; the path and star topologies are examined, the cross and T topologies are examined with respect to every single candidate Steiner point, and the H topology is examined with respect to every pair of candidate Steiner points. The shortest tree seen is returned as the optimal tree. This computation incurs a time complexity of $O(n^2)$, dominated by checking the H topology.

These algorithms are similar to Hakimi's spanning tree enumeration algorithm [12]; however, identifying and examining the various topologies explicitly has tremendous dividends. For example, a straightforward implementation of Hakimi's spanning tree enumeration algorithm using a minimum spanning tree algorithm is many times slower than our explicit enumeration algorithms. For randomly generated instances with 10 rectangular obstacles, Table 2 shows the average running time of the explicit enumeration algorithm versus the average running time of a spanning tree enumeration algorithm using a minimum spanning tree routine. As can be seen from the table, the spanning tree enumeration is far slower than explicit enumeration.

It is possible to perform the case analyses and construct similar explicit enumeration algorithms for exact solution of problem instances with more than four terminals. However, the number of possible topologies increases exponentially, and examining them all rapidly becomes too expensive. For instance, for five terminals there are several hundred topological instances. The explicit enumeration approach might be practically applicable for five terminals, but almost certainly not for six. We recommend a heuristic approach for nets with more than four terminals.

6 Heuristics

Although for nets with more than four terminals exact solution by explicit enumeration is impractical, heuristics can be used to quickly find good solutions for such nets. Given the exact three- and four-terminal algorithms in Section 5, a natural approach is *Steinerization*. In a Steinerization heuristic, portions of a minimum spanning tree that contain a few adjacent terminals are replaced with an optimal Steiner subtree for those terminals. Typically, such heuristics examine subsets of a fixed size, i.e. subsets of size K for some small K. In light of the results in Section 5, we examine heuristics for $K = 3$ and $K = 4$.

The first heuristic, *greedy Steinerization*, starts with a minimum spanning tree of the terminals. It then repeatedly examines vertex subsets of size K that are adjacent in the MST, Steinerizing the one that improves the minimum spanning tree the most. The Steiner points introduced by the Steinerization are candidates for further Steinerization in later iterations. For $K = 3$, greedy Steinerization is an oft-repeated idea whose genesis is unclear. For the standard RST problem, Richards (see Hwang, Richards, and Winter [16]) first investigated 3-Steinerization in this greedy form, and more complex variants appear in Chao and Hsu [3], Lee, Bose, and Hwang [19] and Smith, Lee, and Liebman [24]; these and others are summarized in Hwang, Richards, and Winter [16]. For the OARST problem, greedy 3-Steinerization (henceforth called *G3S*) has time complexity $O(k^2 n)$. For $K = 4$, greedy Steinerization is similar to the algorithm of Beasley [1], though Beasley's algorithm computes a new MST at each iteration rather than locally modifying the current MST. For the OARST problem, greedy 4-Steinerization (henceforth, *G4S*) has time complexity $O(k^3 n^2)$.

We can speed up Steinerization heuristics by a batching trick similar to the heuristic of Hasan, Vijayan, and Wong [14] for the standard RST problem. In their *neighborhood Steinerization* heuristic, each vertex v is assigned a weight that is the amount of improvement over the MST that is gained by Steinerizing v and its neighbors. Since any vertex in a rectilinear minimum spanning tree can have at most 8 neighbors, each Steinerization can be performed in constant time. Since we cannot efficiently Steinerize

k	3	5	7	9	11	13	15
r	1.0	1.5	1.8	2.0	2.1	2.2	2.3

Table 3: Average iteration count for B3S.

large neighborhoods for the OARST problem, in our heuristic the weight of a vertex v is instead the best improvement gained by 3-Steinerizing v and any two of its neighbors.

The heuristic then finds a *maximum-weight independent set (MWIS)* of the tree; this can be computed in $O(k)$ time by dynamic programming. The best 3-neighborhood of each vertex in the MWIS is then Steinerized, and the process is repeated for the new tree produced by replacing each neighborhood with its Steiner subtree. The time complexity of this algorithm, which we call *batched 3-Steinerization (B3S)* is $O(rkn)$, where r is the number of iterations required, which is a function of k. In the worst case, r is equal to k, so the worst-case time complexity is the same as for G3S; however, this bound is quite pessimistic. Table 3 shows the average value of r for various values of k, for randomly generated instances containing 10 rectangular obstacles. Empirically it appears that r is $O(\log k)$, giving B3S a time complexity of $O(nk \log k)$ in practice.

We can also perform another optimization to reduce the running time of Steinerization heuristics. Before computing the Steiner subtree for each subset T of the terminals, perform the reductions described in Section 4 on the escape graph, considering only members of T to be terminals. Since the size K of the subsets considered is small, the reduction in the size of escape graph is typically substantial. Since the number of terminals K is constant, the reductions have time complexity $O(n)$ for each subset. For 3-Steinerization, this is the same as the time complexity of actually Steinerizing the subset, so performing the reductions is not productive. However, for 4-Steinerization, the cost of Steinerizing each subset is $O(n^2)$, so linear-time preprocessing can be effective if it substantially reduces n. For $K = 4$, we expect roughly 80% of the vertices to be eliminated (see Figure 3(b)), and as it turns out, this approach does indeed dramatically improve the running time of G4S—for the 20-terminal instances tested, this optimization improves the average running time by a factor of almost 2.5.

Table 4 shows the result quality (percent improvement over the minimum spanning tree) and runtime for G3S and B3S, and for G4S with the reduction optimization described above, for randomly generated instances containing 10 rectangular obstacles and the indicated numbers of terminals. For the standard RST problem, the average improvement of optimal RSTs over the minimum spanning tree is roughly 12% (see Hwang, Richards, and Winter [16]). For the instances of the OARST problem tested here, this value is between 9.5% and 10.5%. Thus, the improvement values in Table 4 should not be compared with those reported in the literature for standard RST heuristics. The reader should note that B3S produces trees roughly as good as, and sometimes better than, those produced by G3S. In addition, note that G4S consistently produces better trees than G3S, but that the difference in running times is not nearly as pronounced as one would expect from their relative time complexities—in particular, the running time of G4S *decreases* relative to the running time of G3S as k increases.

	G3S		B3S		G4S	
k	Qual.	Time	Qual.	Time	Qual.	Time
4	7.83	0.25	7.83	0.25	8.19	0.40
5	8.58	0.48	8.59	0.46	9.21	0.97
6	8.55	0.81	8.61	0.75	9.12	1.81
7	8.81	1.26	8.82	1.10	9.35	2.94
8	8.44	1.92	8.40	1.56	9.02	4.55
9	8.74	2.75	8.72	2.10	9.33	6.61
10	9.02	4.17	9.03	2.96	9.53	9.82
12	8.72	7.59	8.69	4.78	9.14	17.64
14	8.93	13.63	8.90	7.48	9.40	31.58
16	8.99	22.66	8.95	11.51	9.48	51.33
18	9.03	34.63	9.04	16.15	9.46	78.79
20	9.02	50.21	8.99	21.53	9.43	112.9

Table 4: Average result quality (percent improvement over MST) and running time (in seconds) for the heuristics.

Note that the worst-case ratio of the length of a minimum spanning tree to the length of an optimal Steiner tree (called the *Steiner ratio*) for the OARST problem is 2 (see Appendix A). All three of these heuristics always produce trees at least as short as the MST, and thus produce trees that are no more than twice the length of an optimal tree. In practice, of course, their performance is rarely that bad.

7 Summary

We have proven a theorem (Theorem 1) that reduces any instance of the obstacle-avoiding rectilinear Steiner tree problem to a graph problem, analogous to Hanan's theorem [13] for the standard rectilinear Steiner tree problem. This theorem enables com-

putation of optimal OARSTs in time corresponding to the size of the instance rather than to the routing area. We have also presented algorithms that quickly compute optimal obstacle-avoiding rectilinear Steiner trees for three- and four-terminal nets, and algorithms that compute good heuristic solutions for larger nets. These results demonstrate that the escape graph model is a viable alternative to previous methods for multi-terminal routing in the presence of obstacles. Of course, many interesting avenues of further research remain.

In particular, Guha and Suzuki [11] present algorithms that compute the rectilinear Voronoi diagram for a set of points in the presence of rectangular obstacles, and algorithms that solve a number of problems, including minimum spanning tree, within this domain. By combining techniques such as theirs with our Theorem 1, one might devise improved algorithms for computing exact and approximate OARSTs.

8 Acknowledgements

The authors' work has been supported in part through National Science Foundation grants MIP-9107717 and CDA-8922545 and Virginia CIT award 5-30971. Their support is greatly appreciated. In addition, thanks are due to Hans Bodlaender, Andrew Kahng, and Dana Richards for helpful discussions on various aspects of this paper.

References

[1] J. E. Beasley, *A heuristic for Euclidean and rectilinear Steiner problems*, European Journal of Operational Research, 58 (1992), pp. 284–292.

[2] J. L. Bentley and T. Ottmann, *Algorithms for reporting and counting geometric intersections*, IEEE Transactions on Computers, 28 (1979), pp. 643–647.

[3] T. Chao and Y. Hsu, *Rectilinear Steiner tree construction by local and global refinement*, in Proceedings of the International Conference on Computer-Aided Design, 1990, pp. 432–435.

[4] C. Chiang, M. Sarrafzadeh, and C. K. Wong, *An algorithm for exact rectilinear Steiner trees for switchbox with obstacles*, IEEE Transactions on Circuits and Systems, 39 (1992), pp. 446–455.

[5] J. P. Cohoon and D. S. Richards, *Optimal two-terminal α-β wire routing*, Integration: the VLSI Journal, 6 (1988), pp. 35–57.

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

[7] R. E. Erickson, C. L. Monma, and A. F. Veinott Jr., *Send-and-split method for minimum-concave-cost network flows*, Mathematics of Operations Research, 12 (1987), pp. 634–664.

[8] G. N. Frederickson, *Fast algorithms for shortest paths in planar graphs with applications*, SIAM Journal on Computing, 16 (1987), pp. 1004–1022.

[9] J. L. Ganley and J. P. Cohoon, *A faster dynamic programming algorithm for exact rectilinear Steiner minimal trees*, in Proceedings of the Fourth Great Lakes Symposium on VLSI, 1994.

[10] 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.

[11] S. Guha and I. Suzuki, *Proximity problems and the Voronoi diagram on a rectilinear plane with rectangular obstacles*, in Proceedings of the Thirteenth Conference on Foundations of Software Technology and Theoretical Computer Science, 1993.

[12] S. L. Hakimi, *Steiner's problem in graphs and its implications*, Networks, 1 (1971), pp. 113–133.

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

[14] N. Hasan, G. Vijayan, and C. K. Wong, *A neighborhood improvement algorithm for rectilinear Steiner trees*, in Proceedings of the International Conference on Circuits and Systems, 1990, pp. 2869–2872.

[15] D. W. Hightower, *A solution to the line-routing problem on the continuous plane*, in Proceedings of the Sixth Design Automation Workshop, 1969, pp. 1–24.

[16] F. K. Hwang, D. S. Richards, and P. Winter, *The Steiner Tree Problem*, North-Holland, Amsterdam, Netherlands, 1992.

Figure 5: Instance for which the Steiner ratio is 2.

Theorem 2 *The Steiner ratio for the OARST problem is 2, i.e., over all instances I of the OARST problem, if $MST(I)$ is the length of an MST for I and $OARST(I)$ is the length of an OARST for I, then*

$$\rho = \max_I \frac{MST(I)}{OARST(I)} = 2.$$

Proof: We first show that $\rho \geq 2$. This is accomplished by the example shown in Figure 5. In the figure, each tower is one unit wide and h units tall, and the towers are 2 units apart horizontally. For k terminals, the length of the minimum spanning tree for such an instance is $(k-1)(2h+5)$, and an optimal OARST has length $hk+4k-3$. Thus, for a given k, if h is arbitrarily large, then $\rho = 2(k-1)/k$. If k is large, then the Steiner ratio is very close to 2.

The following well-known argument shows that $\rho \leq 2$. Suppose τ is an optimal OARST for a given set of terminals and obstacles, whose length is d_τ. An Euler tour of τ results in a spanning tree of the terminals whose length is at most $2d_\tau$. Since the length of the minimum spanning tree cannot exceed the length of the tour, the Steiner ratio ρ cannot exceed 2. □

B Terminals on Obstacle Borders

Often, a routing instance will have all its terminals on the borders of obstacles, as when the terminals are the pins of logic cells. A natural question is whether this restriction improves the inherent complexity of the OARST problem, especially since it implies that each terminal must have degree 1—a restriction under which the standard RST problem is solvable in linear time.

Unfortunately, however, the OARST problem remains NP-complete under this restriction. We prove this by transformation from the standard RST problem.

Figure 7: The escape graph for an instance with terminals on the border of the routing region.

for P' must have length at least $20nK + 34n$. From the OARST for P' we may easily construct the RST for P by replacing each widget with a single terminal at its center, and then dividing the x and y coordinates of every point in the tree by $20n$. □

C Terminals on the Border of a Rectangle

Another special case of the OARST problem is when the terminals lie on the border of the routing region. Mirayala, Hashmi, and Sherwani [20] present an exact algorithm for the case where the region contains only one obstacle, and an approximation algorithm for any number of obstacles. Chiang, Sarrafzadeh, and Wong [4] present an exact algorithm for this special case, that runs in time linear in the number of terminals but exponential in the number of obstacles.

As it turns out, this special case is solvable in polynomial time. The escape graph for any routing instance is clearly planar. Furthermore, if the terminals lie on the border of the routing region, then the terminals in the escape graph all lie on the border of the infinite face of the escape graph. Figure 7 shows an instance with the terminals on the border of the routing region, and its escape graph. Erickson, Monma, and Veinott [7] show that the Steiner problem is solvable in polynomial time for such a graph. One way to achieve polynomial-time solution is with a modification of the Dreyfus-Wagner dynamic programming algorithm [6]. The key to the Dreyfus-Wagner algorithm, and the source of its exponential time complexity, is that it examines all possible subsets of the set of terminals. For the case where the graph is planar and all its terminals lie on the border of the infinite face, the Dreyfus-Wagner algorithm need only consider those subsets of the set of terminals that are adjacent along the border of the infinite face. This results in a time complexity

of $O(k^3n^2 + k^2n^2 \log n + n^2)$.

It should be noted that while the Chiang, Sarrafzadeh, and Wong algorithm [4] has exponential time complexity with respect to the number of obstacles, it has *linear* time complexity with respect to the number of terminals. The modified version of Dreyfus-Wagner described above is fully polynomial, but is superlinear in both obstacles and terminals. It would be interesting to devise an algorithm for this special case that is linear in the number of terminals and polynomial in the number of obstacles, or vice-versa.