Routing a Multi-Terminal Critical Net: Steiner Tree Construction in the Presence of Obstacles

Joseph L. Ganley and James P. Cohoon*
Department of Computer Science
University of Virginia
Charlottesville, Virginia 22903
{ganley,cohoon}@virginia.edu

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 experiments demonstrate that the model and algorithms work well in both theory and practice.

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. The technique of *maze routing* [10, 11] optimally routes two-terminal nets, but uses time and space corresponding to 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 work [8] reduced these demands, but did not produce optimal solutions, and still later work [3] performs optimal two-terminal routing in the presence of obstacles, in time corresponding to the size of the routing instance rather than to the routing area.

The problem of optimally routing a multi-terminal net in the presence of obstacles has received substantially less attention. As a result, VLSI designers typically use a multi-terminal variant of a maze routing algorithm, which incurs the same space demands as the two-terminal variety, and usually produces solutions that are far from optimal. Even in the absence of obstacles, multi-terminal routing corresponds to the *rectilinear Steiner tree (RST)* problem, which is NP-complete [5]. 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, as have many efficient polynomial-time heuristics 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 [6] for the standard RST problem, that transforms an instance of the OARST problem to a graph problem whose size is a function of the input size rather than the routing area.

Using the theorem, we present algorithms that efficiently produce *optimal* solutions for nets with three or four terminals, which comprise the majority of multi-terminal routing problems. We also present heuristics that produce good but nonoptimal solutions for nets with five or more terminals. We present empirical evidence that these algorithms are, indeed, effi-

*The authors gratefully acknowledge the support of National Science Foundation grants MIP-9107717 and CDA-8922545 and Virginia CIT award 5-30971.

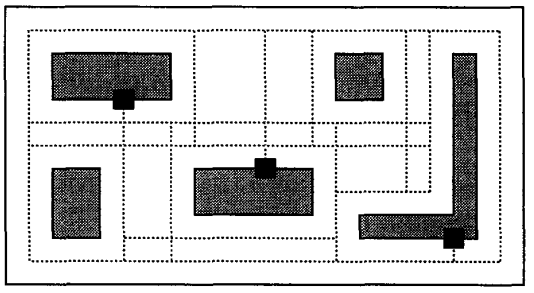

Figure 1: A routing instance and its escape segments.

cient, and that they produce good solutions to multi-terminal OARST problems.

2 The Escape Graph

Figure 1 depicts a custom layout instance and a collection of *escape segments* (shown as dashed lines) for that instance. These escape segments were first used in an optimal two-terminal interconnection technique called *line intersection routing* [3]. The escape segments are a generalization of the line search escape segments used by Hightower [8].

To describe the escape segments, 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*. 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 are the escape segments for the instance depicted.

It has been shown that escape segments suffice for optimal routing of two-terminal nets [3]. We now show that there is an optimal routing for any net with $k > 2$ terminals, that uses only escape segments.

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

Proof: (By contradiction.) 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.

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.

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 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 t. We know there is room to slide, as s is not an escape segment. An escape segment t 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. 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 OARST instance has an optimal solution composed only of escape segments. □

Theorem 1 can be used similarly to Hanan's well-known theorem [6] for the standard RST problem, to construct a graph representation for 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.

3 Escape Graph Generation

The generation of the escape segments from a problem instance is relatively straightforward and is performed by a line-sweep algorithm described by Cohoon and Richards [3]. Generating the escape segments requires $O(m \log m)$ time, where m is the number of obstacle boundary segments.

The intersections of the escape segments are then computed. Since the escape segments are readily 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. The total time complexity of generating the escape graph is thus $O(\max\{n, m \log m\})$.

It is often possible to eliminate many of the vertices in the escape graph, forming a *reduced escape graph*, while still guaranteeing that an optimal solution exists that is constrained to this reduced escape graph. For net distributions found in practice, experiments show that roughly 85% of the vertices can be eliminated on average. The reduced escape graph can be generated from the escape graph in $O(k^2 n)$ time, where k is the number of terminals and n is the number of candidate Steiner points.

4 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 (the vast majority of multi-terminal nets) as efficiently as a typical heuristic solution.

For a three-terminal net, an optimal Steiner tree can have only one of two topologies: a simple path between the terminals, or three terminals connected to a single Steiner point. Thus, an optimal OARST for a three-terminal net may be computed by explicitly checking each of these topologies. The latter topology—where the tree contains a Steiner point—dominates the computation time, and is performed 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 [4].

Similar observations can be made for four-terminal nets. For four terminals, the possible topologies are shown in Figure 2 (terminals are depicted as closed circles, and Steiner points as open circles). An optimal Steiner tree for a four-terminal net can be efficiently computed by explicitly enumerating these topologies, and returning the shortest tree seen. This computation incurs a time complexity of $O(n^2)$, dominated by checking the H topology.

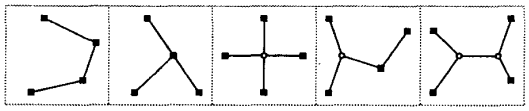

Figure 2: The possible topologies for four terminals.

It is possible to perform the case analyses and construct similar explicit enumeration algorithms for exact solution of instances with more than four terminals, but the complexity increases exponentially with the number of terminals. We recommend a heuristic approach for nets with more than four terminals.

5 Heuristics

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

The first heuristic, *greedy K-Steinerization*, repeatedly examines vertex subsets of size K, 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 the standard RST problem, Richards [9] first investigated 3-Steinerization in this greedy form. For the OARST problem, greedy 3-Steinerization (G3S) has time complexity $O(k^2 n)$. Greedy 4-Steinerization (G4S) 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, G4S has time complexity $O(k^3 n^2)$.

We can speed up Steinerization heuristics by a batching technique similar to the heuristic of Hasan, Vijayan, and Wong [7] for the standard RST prob-

	G3S		B3S		G4S	
k	Qual.	Time	Qual.	Time	Qual.	Time
4	7.83	0.25	7.83	0.25	8.19	0.44
5	8.58	0.48	8.59	0.46	9.21	1.30
6	8.55	0.81	8.61	0.75	9.12	2.86
7	8.81	1.26	8.82	1.10	9.35	5.33
8	8.44	1.92	8.40	1.56	9.02	8.96
9	8.74	2.75	8.72	2.10	9.33	13.91
10	9.02	4.17	9.03	2.96	9.53	21.59
12	8.72	7.59	8.69	4.78	9.14	40.43
14	8.93	13.63	8.90	7.48	9.40	74.04
16	8.99	22.66	8.95	11.51	9.48	119.4
18	9.03	34.63	9.04	16.15	9.46	181.2
20	9.02	50.21	8.99	21.53	9.43	256.1

Table 1: Average result quality (percent improvement over MST) and running time (in seconds) for the heuristics.

lem. 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 an RST can have at most 8 neighbors, each Steinerization can be performed in constant time.

Since we cannot efficiently Steinerize 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 in an MST. The heuristic then finds a *maximum-weight independent set (MWIS)* of the tree, which 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. In the worst case, r is equal to k, so the worst-case time complexity is the same as for G3S; however, empirically it appears that r is $O(\log k)$, so B3S has a time complexity of $O(nk \log k)$ in practice.

Table 1 shows the result quality (percent improvement over the minimum spanning tree) and runtime for G3S, B3S, and G4S, for randomly generated instances containing 10 rectangular obstacles and the indicated numbers of terminals. We have found that the average improvement of optimal OARSTs over the minimum spanning tree is somewhat lower than for the standard RST problem, so the improvement values in Table 1 should not be compared with those reported for standard RST 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. 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.

References

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

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

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

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

[5] M. R. Garey and D. S. Johnson. The rectilinear Steiner tree problem is NP-complete. *SIAM Journal of Applied Mathematics*, 32:826–834, 1977.

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

[7] 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*, pages 2869–2872, 1990.

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

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

[10] C. Y. Lee. An algorithm for path connections and its applications. *IRE Transactions on Electronic Computers*, 10:346–365, 1961.

[11] E. F. Moore. Shortest path through a maze. *Annals of the Computational Laboratory of Harvard University*, 30:285–292, 1959.