Circuit Simulation Based Obstacle-Aware Steiner Routing *

Yiyu Shi, Paul Mesa, Hao Yu and Lei He
Electrical Engineering Dept., University of California, Los Angeles, CA, 90095, USA
{yshi, pmesa, hy255, lhe}@ee.ucla.edu

ABSTRACT

Steiner routing is a fundamental yet NP-hard problem in VLSI design and other research fields. In this paper, we propose to model the routing graph by an RC network with routing terminals as input ports and Hanan nodes as output ports. We show that the faster an output reaches its peak, the higher the possibility for the correspondent Hanan node to be a Steiner point. Iteratively adding one or multiple selected Steiner points to build and improve Steiner trees leads to 1-cktSteiner and Blocked-cktSteiner (in short, B-cktSteiner) algorithms, respectively. When there are no routing obstacles, 1-cktSteiner obtains similar wirelength compared with the best existing algorithm FastSteiner. Both are less than 1% *worse than the exact solution, but 1-cktSteiner is up to* $11.3X$ *faster than FastSteiner. Compared with the fastest existing heuristic FLUTE, B-cktSteiner has similar runtime but up to* 1.9% *shorter wirelength. Different from FastSteiner and FLUTE which are only applicable to non-obstacle cases, 1-cktSteiner and B-cktSteiner can be applied to routing with obstacles with minimal runtime increase. Compared with the best existing obstacle-avoiding algorithm An-OARSMan, 1-cktSteiner has similar runtime and reduces wirelength by* 6.12%*, and B-cktSteiner has an average speedup of* $352X$ *with a similar wirelength.*

Categories and Subject Descriptors: B.7.[Hardware]: – Integrated Circuits–Design Aids

General Terms: Algorithms, Design, Performance

Keywords: Routing, Simulation, RSMT, OARSMT

1. INTRODUCTION

Rectilinear Steiner minimum tree (RSMT) construction is a fundamental research problem in VLSI design. For a given set of terminals, the RSMT problem is to find a set of additional points, Steiner points, such that the rectilinear minimal spanning tree (RMST) connecting all terminals and Steiner points has the minimal length. The RSMT problem is NP-complete. Yet, a few properties have been revealed to help solve this problem: An optimal RSMT can be found in the Hanan grid, which is composed by horizontal and vertical lines from each terminal; Also, at most $n-2$ Steiner points are required to construct an optimal RSMT.

GeoSteiner [1], an exact algorithm with a high complexity, and several heuristics [2–6] have been proposed, all assuming no obstacles for routing. In practice, macro cells, IP blocks and pre-routed nets are considered as obstacles for routing, and obstacle-avoiding RSMT (OARSMT) construction must be studied. Escape graph is often used to convert the OARSMT problem to a RSMT problem on a graph. The distance between two points in the presence of obstacles is calculated based on the obstacle-avoiding shortest path algorithm. [7] presented an exact algorithm for obstacle-avoiding Euclidean Steiner tree construction, but its high complexity prohibits it from practical use. The routing quality of line search heuristics is not good for multiple terminals. Most existing obstacle-avoiding RSMT algorithms use multi-terminal variants of the maze algorithm, with a high space demand but a result far from optimal. Based on track graph reduction and Ant-Colony-Optimization, a very recent work, *An-OARSMan* [8], achieves small wirelength with a reduced but still long runtime.

The existing heuristics for RSMT and OARSMT improves either runtime or wirelength but not both when the number of terminals is large. Ideally, we need an algorithm to achieve the best quality and efficiency of existing work simultaneously. To this end, we propose an algorithm, *cktSteiner*, which simulates the routing problem by circuit behavior. In our algorithm, the routing graph is modeled as an RC mesh. When impulse currents are applied at the terminals, the faster a node reaches its peak voltage, the higher the possibility is for the node to become a Steiner point. Therefore, we can easily select Steiner points from Hanan nodes and build high-quality RSMT and OARSMT efficiently. We call the resulting algorithms *cktSteiner*.

cktSteiner has a few advantageous algorithmic features. It uses numerical circuit simulation (precisely, numerical model order reduction to obtain three poles/residuals) to determine Steiner points, while virtually all existing works use combinatorial algorithms. *cktSteiner* applies to both RSMT and OARSMT with a small runtime difference, but existing RSMT algorithms either can not be extended to the OARSMT problem or suffer a big runtime increase. Because *cktSteiner* simulates routing by circuit behavior, it is a new addition to the existing simulation-based algorithms such as simulated annealing, genetic algorithm, and force-based (placement) algorithm that have been successfully used in VLSI design. In addition, *cktSteiner* obtains the best wirelength in the shortest runtime compared with the existing algorithms [2-8].

The remainder of the paper is organized as follows: Section 2 describes the problem formulation, Section 3 introduces *cktSteiner* algorithms and Section 4 presents experimental results. We conclude in Section 5. A complete version of this paper is available at *http://eda.ee.ucla.edu*.

2. PROBLEM FORMULATION

In this paper we start from the routing model as in [9] and tessellate the routing area into rectangular partitions as *global tiles*, and pins within the same global tile are mapped into the center of it. The routing plane can be formally modeled by an undirected graph $G_h(V, E)$, namely the *Hanan grid*, where each vertex $v \in V$ represents a global tile, and each edge $e \in E$ represents the routing area between two adjacent tiles. An example of the Hanan grid is shown in Figure 1 (a). To consider the impact of obstacles such as hard macros or pre-routed nets on routing, the routing graph G_e should be constructed by intersecting lines from vertices as well as the edges of the obstacles. We call the re-

*This paper is partially supported by NSF CAREER award CCR-0093273/0401682. Address comments to lhe@ee.ucla.edu.

DAC 2006, July 24–28, 2006, San Francisco, California, USA.

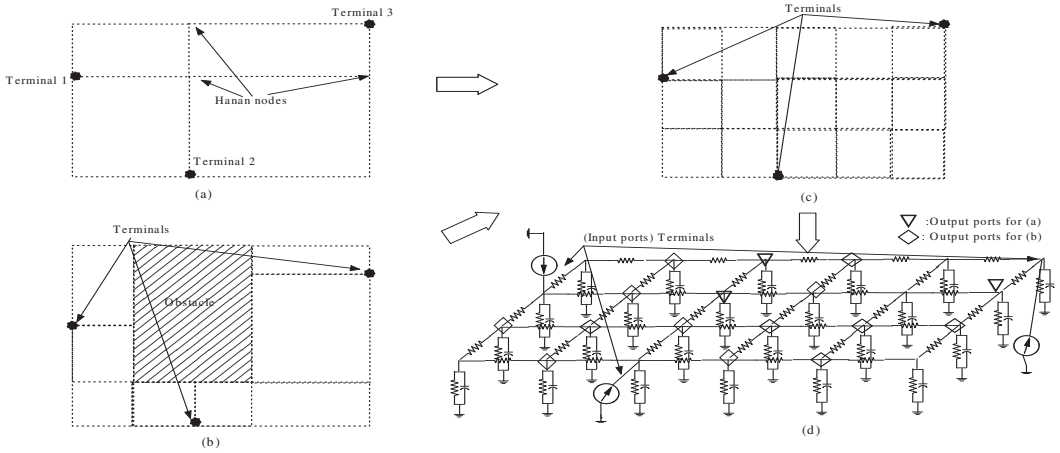

Figure 1: (a) Hanan grid with three terminals. (b) Escape graph with the same three terminals and one obstacle. (c) The corresponding routing graph. (d) The corresponding RC mesh

sulting obstacle-avoiding routing graph an *escape graph* [10]. An example of an escape graph is shown in Figure 1 (b).

Without the loss of generality, in this paper, we use a uniform G, or global routing graph (GRG), which is fine enough so that all the Hanan nodes or the nodes of the escape graph are located on the nodes of it, i.e., the Hanan grid G_h or the escape graph G_e are the subgraphs of G. It is obvious that any Steiner tree constructed in the Hanan grid or in the escape graph can also be found in our routing graph. Such a routing graph is shown in Figure 1 (c), which works for both (a) and (b). By introducing this routing graph, routing with or without obstacles are indistinguishable from each other except that the distance between two nodes should be the obstacle-avoiding distance in the cases of obstacle-avoiding routing.

With respect to the above discussions, we formulate the following problem:

FORMULATION 1. *Given a routing graph G as constructed above with an embedded multi-terminal net, find a set of Steiner points in G such that the resulting Steiner routing of the multi-terminal net has minimum routing wirelength.*

3. PROPERTIES AND ALGORITHMS

3.1 Circuit Model and its Implication

To map GRG into an RC mesh circuit model, we model each edge of GRG with a unit resistor, and connect each vertex of GRG to ground via a unit capacitor and a unit resistor in parallel. Terminals are modeled as input ports, each with a unit impulse current source. The Hanan nodes or the nodes of the escape graph are modeled as output ports. Such a circuit model is illustrated in Figure 1 (d). Note that when there are obstacles, we still keep the resistors and capacitors in the obstacle area.

With a unit impulse current source at each terminal at time $t = 0$, the signals start to propagate until steady state is reached. It takes a finite time for the signal to propagate throughout the mesh and to charge the capacitors. Then the signal at one node would decay through the DC path of the grounded resistors. We define *peak time* as the time for the voltage response to reach its maximum value.

Take Figure 2 as an example. Shown in (a)is a net with three terminals (labeled with circles) embedded within a 11×10 routing graph G, where the corresponding Hanan nodes are also marked (with triangles). For the ease of presentation, we assign a label to each node in G. The voltage responses at the Hanan nodes (vertices #39, #61 and #63) are shown in Figure 2 (b). The peak time at vertex #61 is smaller than those other two nodes, and vertex #61 is actually the Steiner point for the 3-terminal set.

The above example implies that the faster a voltage response reaches its peak, the higher the probability is for it to become a Steiner point. This can be explained as follows: in general, Steiner points tend to have small distances to all nearby terminals. Similarly, in the mesh, the time constant between two points is nearly proportional to their distance. Therefore, the more likely a node is a Steiner point, the smaller the weighted distance is from this node to the terminals, in turn the smaller the RC time constant is for the node, and finally the smaller its peak time is.

Similar phenomenons can be observed in other cases, too. We randomly generate 20 test cases with 100 terminals, and construct the optimal RSMT by *GeoSteiner* [1] to get all the Steiner points. We order all the Hanan nodes in sequence with increasing peak times and calculate the probability for each Hanan node to become a Steiner point in the optimal solutions. The results are shown in Figure 3. The x-axis is the order in the sequence, and the y-axis is the normalized probability. Generally, the probability decreases when the order of the Hanan node in the sequence decreases and the peak time increases.

Based on these results, we have the following observation:

Observation 1. *A Hanan node is more likely to become a Steiner point when the voltage response of the corresponding node in the RC mesh reaches its peak earlier.*

Algorithm 1 *1-cktSteiner*

OUTPUT: Steiner point set S, $RSMT = MST\{S \cup T\}$;
Initialization: Steiner point set $S = \Phi$;
Initialization: $l_0 = MST(T)$, $flag = 0$;
while There are less than $n-2$ nodes in S and $E \neq \Phi$ and $flag < n/8$ **do**
  while The 1st node A in E is in the current tree **do**
    Remove A from E;
  end while
  Select the first node A in E;
  $l_1 = MST\{S \cup T \cup A\}$;
  if $l_1 < l_0$ **then**
    $S = S \cup A$;
    $l_0 = l_1$;
    $flag = 0$;
  else
    $flag = flag + 1$;
  end if
  Remove A from E;
end while

3.2 Steiner Tree Construction Algorithms

Based on Observation 1, we determine Steiner points based on voltage response in the RC mesh for the routing graph.

The voltage response at one node of an RC network can be expressed as:

$$v(t) = \sum_{i=1}^{q} r_i e^{\frac{t}{\lambda_i}}, \quad (1)$$

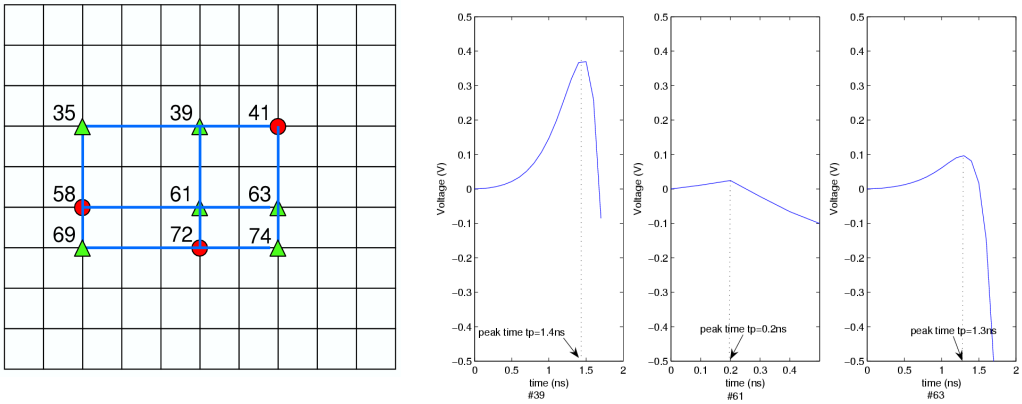

Figure 2: (a) Illustration of a routing graph with a three-terminal net (circle) and its corresponding Hanan nodes (triangle). (b) Time domain responses at the nodes #39, #61, and #63 in (a). Note that for #61 a different x-axis range is used. It reaches voltage peak quicker than others and is the Steiner point.

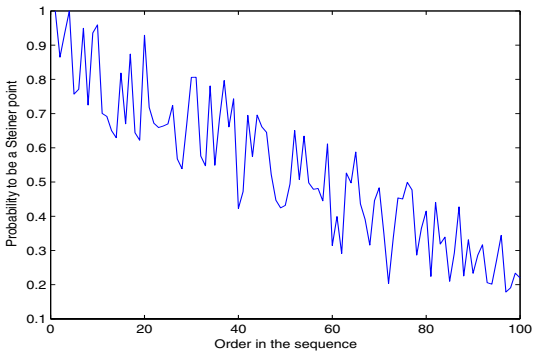

Figure 3: The probability for the Hanan nodes to become a Steiner point with respect to the order in the sequence for twenty 100-terminal test cases.

where n is the pole number, λ_i are the poles, and r_i are their corresponding residuals. Usually, only the first two and three poles are dominant for a linear RC circuit. Therefore, third order approximation ($q = 3$) of (1)is used in this paper. By taking the derivative with respect to time in (1) with the third-order approximation, we can calculate the peak time, t_{peak}, as

$$\sum_{i=1}^{3} \frac{r_i}{\lambda_i} e^{\frac{t_{peak}}{\lambda_i}} = 0 \qquad (2)$$

Then we build a sorted and pruned Hanan nodes sequence in the ascending order according to their peak times. Because at most $n-2$ points need to be added into the RSMT, we can use our ordered Hanan nodes sequence to construct the RSMT based on *1-Steiner* heuristic. Our algorithms are faster than other *1-Steiner* based heuristics in the sense that it does not need to employ a special algorithm to select the Steiner points during tree construction.

We first calculate the wirelength of the MST given the set of input terminals. Then *iterated 1-Steiner* idea can be employed. We iteratively add one Hanan node according to its order in the sequence. If one Hanan node is already in the tree we construct, we skip it. Then we compare the wirelength of the new MST with the previous MST. If the new wirelength is shorter, then the node is selected. We continue this step until we have added $n-2$ Steiner points (which is the maximum possible value) or we have examined a user-defined consecutive number (which is $n/8$ in this paper) of Hanan nodes that fail to decrease the wirelength. The *1-cktSteiner* algorithm is summarized in Algorithm 1.

In general, more than one Steiner node can be added each time for the algorithm in Algorithm 1. We call the vertices to be added simultaneously as a block of vertices. If the total wirelength using a block is reduced, then we take all the vertices in the block as Steiner points, otherwise, we check the vertices in the block one-by-one. In this case, *1-cktSteiner* algorithm becomes a block based algorithm, and the number of Steiner points in a block is called *block size*. In Figure 4, we study the interaction between wirelength, runtime and block size. When the block size increases from 1 to 10, clearly the runtime decreases but the wirelength increases. It is easy to see that block size is an effective knob for trade-off between wirelength and runtime. To accommodate different numbers of terminals, we can use a self-adjustable block size. In experiments we find that given the total terminal number n, setting block size $B \in (\frac{n-2}{16}, \frac{n-2}{4})$ can result in a good balance between wirelength and runtime, which leads to the *B-cktSteiner*.

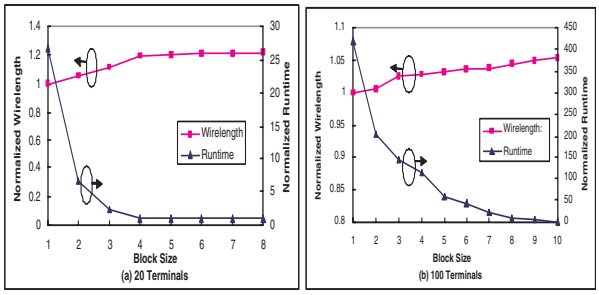

Figure 4: Tradeoff between wirelength and runtime in (a) 20 terminal case and (b) 100 terminal case. The wirelength and runtime are normalized.

4. EXPERIMENTS AND DISCUSSIONS

We experiment on a few groups of test cases, each group for a selected number of terminals. We generate twenty test cases for each group, with terminals randomly placed in a routing plane with 1000×1000 grid. We report the average wirelength and runtime for each group. All experiments are conducted on a UNIX workstation with 1.9GHz P4 processor and 2GB RAM.

We implement the circuit construction and simulation in MATLAB and the *cktSteiner* tree construction part in C language. For experiments below, we always use $B = \frac{n-2}{8}$ for *B-cktSteiner*. Intuitively, if we divide GRG (general routing graph) into a finer grid, we may simulate the routing plane more accurately and obtain shorter wirelength but longer runtime. Figure 5 illustrates how the grid granularity influences wirelength and runtime. We use a test case with 20 terminals routed by *1-cktSteiner*. When the grid becomes finer, runtime increases and wirelength reduces. A nice tradeoff between runtime and wirelength is achieved by a 4X finer grid, where wirelength is reduced by 6% and runtime in-

Terminal #	Wirelength					Runtime (ms)				
	Geo	FastSteiner	1-ckt	Flute	B-ckt	Geo	FastSteiner	1-ckt	Flute	B-ckt
5	9	9	9	9	9	3.05	0.23	0.06	0.0007	0.0006
10	27	27	27	27	27	3.63	0.32	0.09	0.008	0.009
20	77	78	78	79	80	14.4	1.8	0.80	0.043	0.038
50	290	291	292	303	305	38.6	8.1	1.53	0.18	0.23
100	811	821	819	862	848	298	15	3.12	0.47	0.62
500	8305	8377	8395	9032	8861	12600	140	12.4	3.97	5.31
Average	1.000	1.006	1.007	1.037	1.034	1.000	0.093	0.025	0.002	0.002

Table 1: Comparison between Geo-Steiner, FastSteiner, 1-cktSteiner, FLUTE and B-cktSteiner.

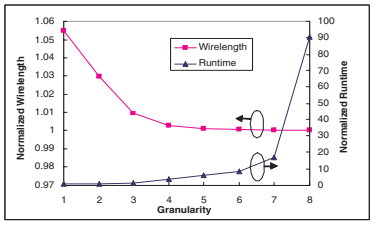

Figure 5: Normalized runtime and wirelength with respect to RC mesh granularity. The test case has 20 terminals and is routed by *1-cktSteiner*.

T #	O #	Wirelength			Runtime (s)		
		A-O	1-ckt	B-ckt	A-O	1-ckt	B-ckt
5	3	4380	4380	4620	0.02	0.06	0.00001
10	9	26990	26980	27450	0.07	0.24	0.0009
20	10	43630	41270	45820	0.24	0.49	0.03
50	10	53260	50710	53770	2.58	4.17	0.98
100	10	80040	76380	81340	26.9	32.5	2.37
500	20	200360	188090	203240	1660	1082	109
Average		1.000	0.965	1.027	1.000	1.651	0.089

Table 2: Comparison between An-OARSMan, 1-cktSteiner and B-cktSteiner for various terminal (T) and obstacle number (O).

creases by $3X$ compared to using the original grid (equivalent to GRG). A similar tradeoff has been observed for other test cases as well. Therefore, we use a 4X finer grid in all experiments below.

Table 1 presents experiments for obstacle-free routing. We first compare *1-cktSteiner* with the exact solution of *GeoSteiner* [1], and *FastSteiner* which generates the shortest wirelength among existing heuristics. Both *1-cktSteiner* and *FastSteiner* [5] are about less than 1% worse than *GeoSteiner*. *1-cktSteiner* is on average $5.2X$ faster ($11.3X$ faster for the largest example) than *FastSteiner*, which in turn is on average $208X$ faster than *GeoSteiner*. Table 1 also compares *B-cktSteiner* with *FLUTE* [6], the fastest algorithm among existing heuristics. *B-cktSteiner* achieves up to 1.9% wirelength reduction with a similar runtime compared to *FLUTE*. On average, *B-cktSteiner* is 3.4% worse than the exact solution, and *FLUTE* is 3.7% worse. Compared to *1-cktSteiner*, *B-cktSteiner* obtains similar wirelength for up to 20 terminals but 2.7% longer wirelength for larger numbers of terminals, however the runtime of *B-cktSteiner* is $24X$ smaller.

Table 2 presents experiments for routing with obstacles. Compared to *An-OARSMan* [8], the best existing heuristic for obstacle-avoiding routing, *1-cktSteiner* reduces wirelength by up to 6.12% for large test cases at a similar runtime, and *B-cktSteiner* has an average speedup of $352X$ with wirelength similar to that produced by *An-OARSMan*.

We also perform a study on the relationship between the number of terminals (n) and the number of Steiner points (q) used by *1-cktSteiner*. For each terminal number, we use 10 randomly generated test cases for both the obstacle-free routing and the obstacle-avoiding routing. The result is shown in Figure 6. Clearly, the maximum number of Steiner points required is less than $n-2$, and in most cases, about $\frac{n}{2}$ Steiner points are required for both RSMT and OARSMT. This observation will be verified and may be used to develop more efficient algorithms in the future.

5. CONCLUSIONS AND DISCUSSIONS

Using RC network to simulate routing, we have proposed a circuit simulation based Steiner routing algorithm called *1-cktSteiner*, and a faster version *B-cktSteiner* algorithm. When constructing RSMT without obstacles, *1-cktSteiner* obtains similar length but

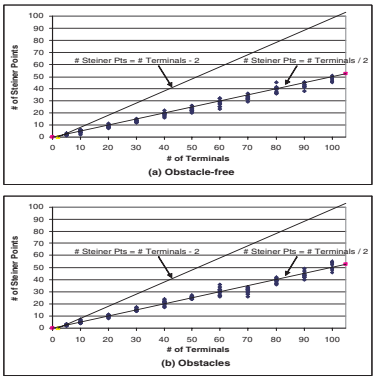

Figure 6: The relationship between the number of terminals n and the number of Steiner points q required using *1-cktSteiner*: (a) obstacle-free routing and (b) obstacle-avoiding routing

runs $11.3X$ faster compared to *FastSteiner*, the existing algorithm with the minimum wirelength. *B-cktSteiner* reduces wirelength by up to 1.9% at similar runtime compared with *FLUTE*, the existing most efficient algorithm. In addition, our algorithms can deal with obstacle-avoiding cases at a similar runtime compared with obstacle-free cases. *1-cktSteiner* reduces up to 6.12% wirelength and runs $352X$ faster compared with *An-OARSMan*, the existing best algorithm for obstacle-avoiding routing.

6. REFERENCES

[1] D. W. Warme and et al, "Geosteiner 3.1. package,"
[2] A. B. Kahng and G. Robins, *On Optimal Interconnections for VLSI*. Boston: Kluwer Academic Publishers, 1995.
[3] I. Mandoiuand and et al, "A new heuristic for rectilinear steiner trees," *IEEE TCAD*, 2000.
[4] H. Zhou, "Efficient steiner tree construction based on spanning graphs," in *ISPD*, 2003.
[5] A. B. Kahng and et al, "Highly scalable algorithms for rectilinear and octilinear steiner trees," in *ASPDAC*, 2003.
[6] C. Chu and Y.-C. Wong, "Fast and accurate rectilinear steiner minimal tree algorithm for vlsi design," in *ISPD*, 2005.
[7] M. Zachariasen and P. Winter, "Obstacle-avoiding euclidean steiner trees in the plane: an exact algorithm," in *extended abstract presented at the Workshop on Algorithm Engineering and Experimentation*, 1999.
[8] Y. Hu and et al, "An-oarsman: Obstacle-avoiding routing tree construction with good length performance," in *ASPDAC*, 2005.
[9] C. Albrecht, "Provably good global routing by a new approximation algorithm for multicommodity flow," in *ISPD*, 2000.
[10] J. L. Ganley and J. P. Cohoon, "Routing a multi-terminal critical net: Steiner tree construction in the presence of obstacles," in *ISCAS*, 1994.