Circuit Simulated Obstacle-Aware Steiner Routing

Yiyu Shi, Paul Mesa, Hao Yu and Lei He
University of California, Los Angeles

This paper develops circuit simulated routing algorithms. We model the routing graph by an RC network with terminals as inputs, and show that the faster an output reaches its peak, the higher the possibility is for the corresponding Hanan or escape node to become a Steiner point. This enables us to select Steiner points and then apply any minimum spanning tree algorithm to obtain obstacle-free or obstacle-aware Steiner routing. Compared with the existing algorithms, our algorithms have significant gain on either wirelength or runtime for obstacle-free routing, and have significant gain on both wirelength and runtime for obstacle-aware routing.

Categories and Subject Descriptors: B.7.2 [**Hardware**]: Integrated Circuits—*Design Aids*

General Terms: Algorithms, Design, Performance

Additional Key Words and Phrases: 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, i.e. 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 [Garey and Johnson 1977]. 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 [Hwang et al. 1992].

GeoSteiner [Warme and et al] is an exact algorithm with a high complexity. The following heuristics have been proposed to improve algorithm efficiency: 1-Steiner [Kahng and Robins 1995] iteratively adds one Steiner point each time to reduce wirelength. A primal-dual approach based on 1-Steiner is also proposed[Mandoiuand et al. 2000]. Recent work to further reduce runtime complexity includes follows: A spanning graph based $O(nlogn)$ heuristic was proposed in [Zhou 2003]. It uses spanning graph to help both generating the initial spanning tree as well as finding good candidates for the edge substitution. A batched greedy triple based $O(nlog^2n)$

Author's address: Electrical Engineering Department, University of California, Los Angeles, CA, 90095.
Some initial results of this paper are published at DAC'06[Shi et al. 2006]. This paper is partially supported by NSF CAREER award CCR-0093273/0401682. Address comments to lhe@ee.ucla.edu.

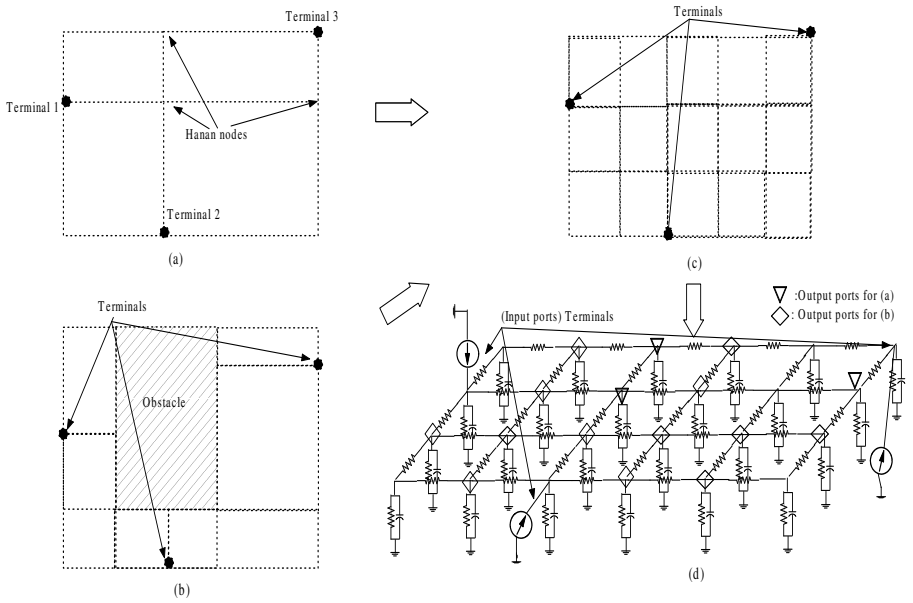

Fig. 1. (a) Hanan grid with three terminals. (b) Escape graph with the same three terminals and one obstacle. (c) The corresponding routing graph. (d) The corresponding RC mesh

heuristic *FastSteiner* [Kahng et al. 2003] was also proposed, which is based on a batched version of the greedy triple contraction algorithm of Zelikovsky [Zelikovsky 1993]. The look-up-table based heuristic *FLUTE* [Chu and Wong 2005] is another fast and accurate technique to perform rectilinear Steiner minimal tree (RSMT) construction. There is a user-defined parameter to control the tradeoff between accuracy and runtime. FLUTE is optimal and extremely fast for nets up to degree 9. As it handles low degree nets particularly well, it is most suitable for VLSI applications in which most nets have a degree 30 or less.

Aforementioned approaches all assume no obstacles for routing. In practice, macro cells, IP blocks and pre-routed nets are considered as obstacles for routing. Therefore obstacle-avoiding RSMT (OARSMT) construction must be studied. Escape graph [Ganley and Cohoon 1994] 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 [Zheng et al. 1996]. [Zachariasen and Winter 1999] presented an exact algorithm for obstacle-avoiding Euclidean Steiner tree construction, but its high complexity prohibits it from practical use. As opposed to the case for RSMT with no obstacles, few heuristics have been proposed for OARSMT due to the difficulty in handling obstacles. Line search heurisitic was introduced in [Hightower 1969] and [Mikami and Tabuchi 1968], but the routing quality is not good for multiple terminals. Most existing obstacle-avoiding RSMT algorithms (e.g., [Akers 1967; Soukup 1978; Hadlock 1977; Rubin 1974]) use multi-terminal variants of the maze algorithm, with a high space demand but a result far from optimal. *An-OARSMan* was proposed recently [Hu et al. 2005] based on ant colony optimization. A greedy

obstacle penalty distance (OP-distance) local heuristic was used in the algorithm and performed on the track graph. It achieves short wirelength with reduced but still long runtime.

The quality of an RSMT is measured by the so called Steiner ratio [Hwang et al. 1992], the ratio of the Steiner tree length over the MST length. The existing heuristics for RSMT and OARSMT improves either runtime (e.g., [Chu and Wong 2005]) or quality (e.g., [Zhou 2003; Hu et al. 2005]), but not both for a large number of terminals. 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, we use dominant voltage responses at Hanan nodes to decide Steiner points. 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*. Similar to *1-Steiner*, we develop both *1-cktSteiner* and *B-cktSteiner* algorithms, depending on whether one or multiple Steiner points are iteratively added to build or improve the routing.

cktSteiner has a few advantageous algorithmic features. It uses numerical circuit simulation to determine Steiner points, while virtually all existing approaches 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.

We have also observed exciting experimental results. When there are no routing obstacles, *1-cktSteiner* obtains similar wirelength compared with the best existing heuristic *FastSteiner* [Kahng et al. 2003]. 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 a similar runtime but up to 1.9% shorter wirelength. Without modifications, *1-cktSteiner* and *B-cktSteiner* can directly be applied to routing with obstacles and the runtime increase is minimal. Compared with the best existing obstacle-avoiding *An-OARSMAN* algorithm, *1-cktSteiner* has similar runtimes and reduces wirelength by 6.12% and *B-cktSteiner* has an average speedup of $357X$ with similar wirelength. Compared with Magma routing package [mag] containing eight routing algorithms, *1-cktSteiner* reduces the chip level total wirelength by up to 1.23% and reduces net-based wirelength by up to 8.15%.

The remainder of the paper is organized as follows: Section 2 describes the problem modeling and validates the key observation upon which our cktSteiner algorithm is proposed. Section 3 introduces the *cktSteiner* algorithm and speedup techniques. Section 4 presents experimental results. Section 5 concludes the paper.

2. CIRCUIT MODEL FOR ROUTING

In this section, we first show how a routing graph can be mapped into an RC circuit. Then the relationship between the Steiner points and the time domain responses for the RC circuit is revealed, which is the basis of the *cktSteiner* algorithm.

2.1 Problem Formulation

In this paper we start with the routing model as in [Albrecht 2000] and tessellate the routing area into rectangular partitions as *global tiles*, and pins within the same global tile are mapped into the center of the tile. 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 routing graph an *escape graph* [Ganley and Cohoon 1994]. An example of an escape graph is shown in Figure 1 (b).

Without 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 applies to 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.*

2.2 Circuit Model and Its Implication

To map the GRG to an RC mesh circuit model, we model each edge of the GRG with a unit resistor, and connect each vertex of the 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. [1]

With a unit impulse current source at each terminal at time $t = 0$, the signals start to propagate until the 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 starts to decay through the DC path of the grounded resistors.

[1] Therefore, the circuit model is independent of obstacles. This enables us to pre-calculate circuit behavior and apply it to nets with different obstacles, as described in Section 3.2.

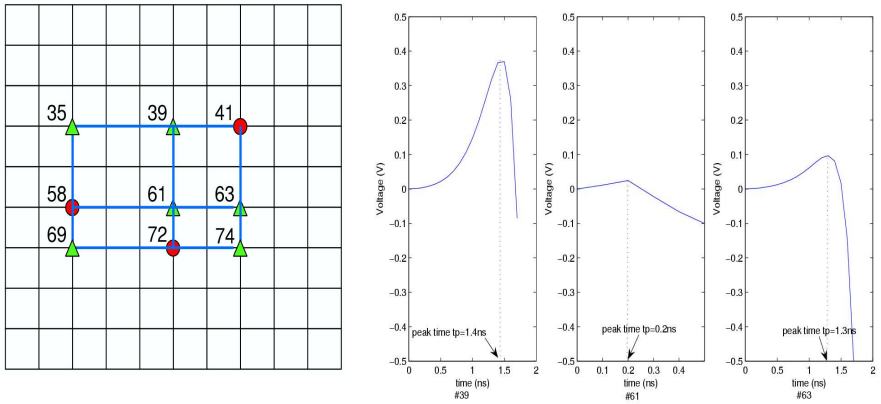

Fig. 2. (a) Illustration of a routing graph with a three-terminal net (circle) and its corresponding Hanan nodes (triangle). (b) Time domain responses at the nodes #39, #61, and #63 in (a). Note that for y-axis, a logarithm scale is used. For node #61 a different x-axis range is used. It reaches voltage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

We define *peak time* as the time for the voltage response to reach its last peak value.

Take Figure 2 as an example. Shown in (a) is a net with three terminals (labeled with circles) embedded within a 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 at the other two nodes, and vertex #61 is the Steiner point for the 3-terminal set.

In the above example, the fastest voltage response indicates the Steiner point. Similar phenomenons can be observed in other cases, too. We randomly generate 20 testcases with 100 terminals, and construct the optimal RSMT by *GeoSteiner* [Warme and et al] to get all the Steiner points. We sort all the Hanan nodes in sequence with increasing peak times and calculate the probability for Hanan node with a given order in the sequence to become a Steiner point in one of these 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. The probability decreases when the order of the Hanan node in the sequence decreases and the peak time increases. One can conclude:

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.*

Note that RSMT is not necessarily unique for a multi-pin net. For example, we might find that in fact nodes 1, 3 and 7 form an optimal RSMT; while nodes 2, 4, 6 form another optimal one. We consider multiple optimal solutions in our tree construction and the probability calculation.

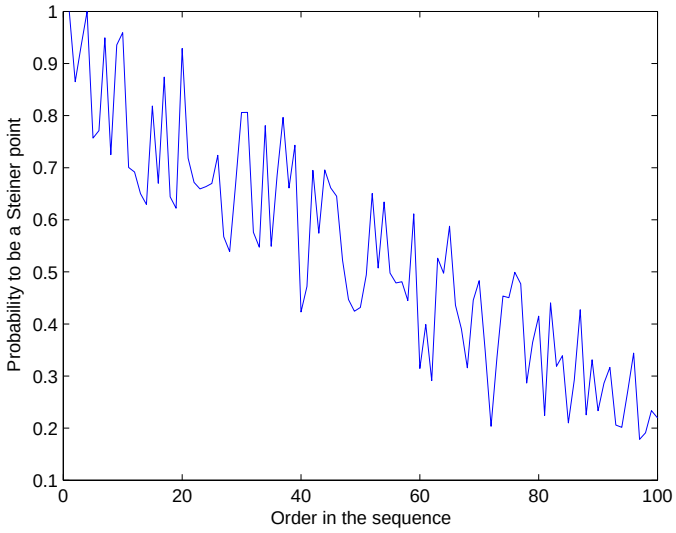

Fig. 3. The probability for the Hanan nodes to become a Steiner point with respect to the order in the sequence for twenty 100-terminal test cases.

2.3 Theoretical Justification

In this section, we theoretically justify Observation 1 by giving an exact proof for 3-terminal cases followed by a qualitative extension to cases with more terminals.

Given a three-terminal net, we label the rectilinear distance between node A and terminal #1 as r_1, and similar similarly define r_2 and r_3. Then the RSMT problem becomes

$$\min_{p} \quad r_1 + r_2 + r_3 \tag{1}$$
$$s.t. \quad p \in \text{the set of Hanan nodes}$$

If the vertex p that minimizes (1) is one of the three terminals, then no Steiner points need to be added and an example for such case is shown in Figure 4(a). Otherwise, the Steiner point p is added to achieve the minimum wirelength, as shown in Figure 4 (b).

Now we prove that the vertex p that minimizes (1) is exactly the vertex that has the minimum peak time t_{peak}. This is done by demonstrating that the objective

$$\min_{p} \quad t_{peak} \tag{2}$$

is a good approximation of (1),

The effective resistance $R_{x,y}$ between two vertices $(0,0)$ and (x,y) in a uniform resistor mesh can be calculated from the following formula [Atkinson and van Steenwijk 1999]:

$$R_{x,y} = \frac{1}{\pi}\int_0^{\pi} \frac{d\beta}{sinh|\alpha|}[1 - e^{-|n\alpha|}cos(p\beta)], \tag{3}$$

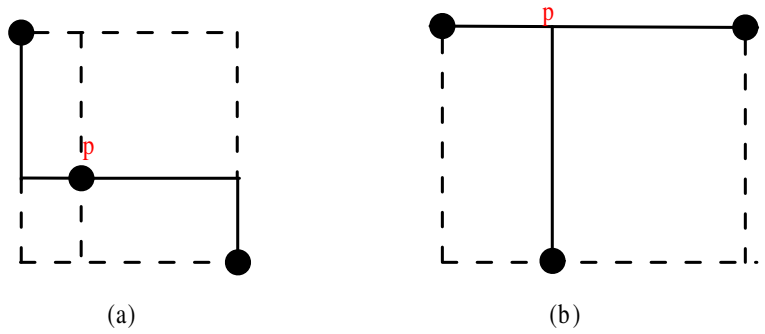

Fig. 4. Two different cases for three-terminal nets: (a) no Steiner points need to be added and (b) one Steiner point is added.

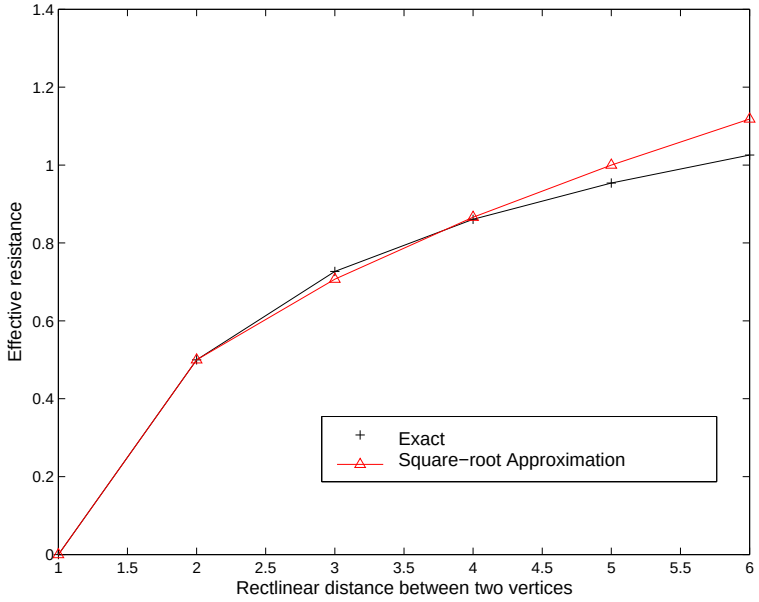

Fig. 5. The relationship between effective resistance and the rectilinear distance of two vertices, and its square-root approximation.

where α and β are complex numbers satisfying

$$cos\alpha + cos\beta = 2. \tag{4}$$

(3) can be well approximated by

$$R_{x,y} = \frac{1}{2}\sqrt{r}, \tag{5}$$

where r is the rectilinear distance between the two vertices. This is demonstrated by Figure 5.

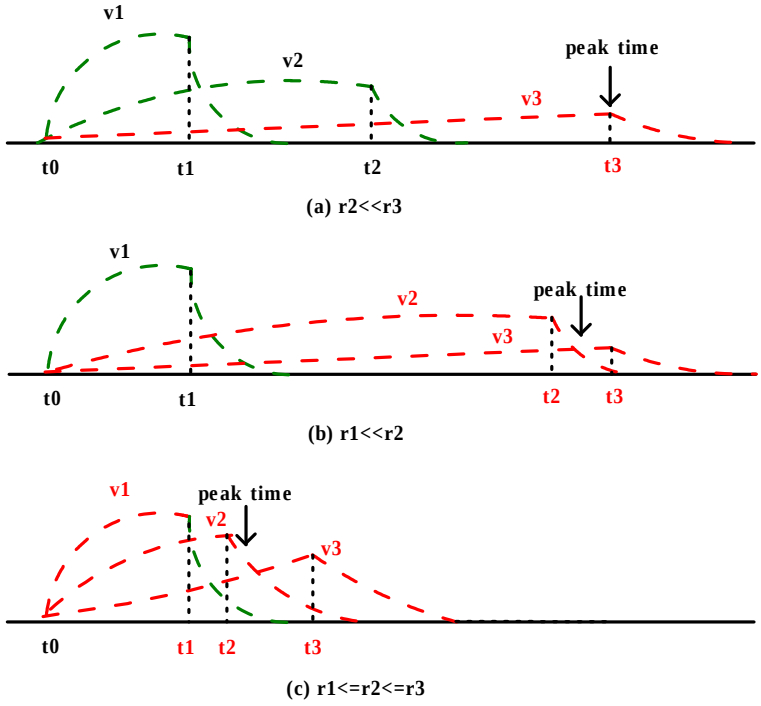

Fig. 6. The peak time w.r.t. to the delay to each of the three terminals in three different cases.

A similar result can be derived for the effective capacitance. Therefore, we can compute the time constant between any two vertices from the lumped RC model:

$$t_{peak} \propto RC \propto \sqrt{r} \times \sqrt{r} = r, \tag{6}$$

which indicates that the delay is linearly proportional to the rectilinear distance. We denote the constant coefficient as k, i.e., $t_{peak} = kr$.

With this important result, we proceed to show that (3) approximates (1) well in different cases. Without loss of generality, we assume $r_1 \leq r_2 \leq r_3$. We plot the waveforms for the three terminals separately in Figure 6: V_1 is generated by keeping the current source at the first terminal and removing the sources at the second and the third terminals. V_2 is computed by only keeping the source at the second terminal and V_3 is computed by only keeping the source at the third terminal. The real waveform at this node is the superposition of the three waveforms plotted. We discuss the following three cases:

Case 1: $r_2 \ll r_3$ (Figure 6(a))

As the figure clearly states, the peak time for the waveform, after superposing the waveforms from three terminals, is mainly decided by $t_3 = kr_3$. Therefore, (3) becomes

$$\min_{p} \quad kr_3. \tag{7}$$

Since $r_2 \ll r_3$, (1) becomes

$$\min_p \quad r_3. \tag{8}$$

Case 2: $r_1 \ll r_2 \leq r_3$; r_2 and r_3 are close. (Figure 6(b))

As the figure clearly states, the peak time for the superposed waveform is approximately $(t_2 + t_3)/2 = k(r_2 + r_3)/2$. Therefore, (3) becomes

$$\min_p \quad k(r_2 + r_3)/2. \tag{9}$$

Since $r_2 \ll r_3$, (1) becomes

$$\min_p \quad r_2 + r_3. \tag{10}$$

Case 3: $r_1 \leq r_2 \leq r_3$; r_1, r_2 and r_3 are close. (Figure 6 (c))

As the figure clearly states, the peak time for the superposed waveform is approximately $(t_1 + t_2 + t_3)/3 = k(r_1 + r_2 + r_3)/2$. Therefore, (3) becomes

$$\min_p \quad k(r_1 + r_2 + r_3)/3. \tag{11}$$

Obviously, (1) and (11) are the same.

In conclusion, (3) approximates (1) well for all 3-terminal nets. Accordingly, our observation 1 is valid for all 3-terminal nets.

In general, for nets with more terminals, Observation 1 can be explained as follows: 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.

Due to the above-explained nature of our algorithm, it can work well for most nets. However, for those nets where the pins are extremely unevenly distributed (some terminals of the nets are far awary), the accuracy of our algorithm will be impacted. This is because the impacts of those terminals are too weak to influence the location of the Steiner points. As a remedy, for those cases we scale the magnitude of the input current soruce proportional to the sum of its distances to all the other terminals. To further improve our algorithm by scaling the magnitudes of the input sources in general cases is still under research.

3. CKTSTEINER ALGORITHMS

In this section, we propose our circuit simulation based algorithm to construct the RSMT for the given set of terminals. We map the global routing graph (GRG) to an uniform RC mesh and add impulse current sources at the terminals. Then Observation 1 is used to select the potential Steiner points. We want to emphasize that *once the potential Steiner points are known, any minimum spanning tree (MST) algorithm can be used to construct the RSMT.*

3.1 Steiner Tree Construction Algorithm

We first build the circuit model for the routing graph, which has been discussed in detail in Section 2: Each edge of the routing graph is replaced with a uniform

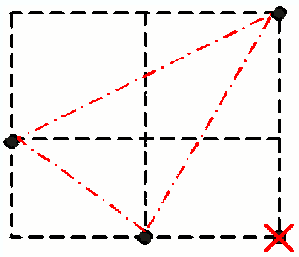

Fig. 7. A Hanan node must reside inside the convex boundary defined by all terminals to become a possible Steiner point and an output port.

resistor, and each node is connected to the ground via a parallel resistor and capacitor. Then an impulse current source is added at each terminal. Note that not all but only these Hanan nodes satisfying the following constraint are considered as output ports and potential Steiner points:

For obstacle-free routing, only Hanan nodes inside the convex boundary defined by all terminals can become Steiner points, as shown in 7.

After the circuit is constructed, we can simulate it and build a sorted Hanan nodes sequence in the ascending order according to their peak times (i.e.,in the descending order of the likelihood to be a Steiner point). Because at most $n-2$ points need to be added into an RSMT [Hwang et al. 1992], we can use our ordered Hanan nodes sequence to construct the RSMT based on the *1-Steiner* heuristic. Our algorithm is 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.

In detail, 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. Note that the MST can be constructed incrementally [Kahng and Robins 1992]. 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 Figure 8.

We use a testcase with four terminals $\{W, X, Y, Z\}$ as shown in Figure 9 to demonstrate the procedure of our algorithm. According to the constraint for the Steiner points, only the node labeled with a, b and c can become Steiner points. We simulate the corresponding circuits and find the peak time for those three points are $13.5ns$, $21.5ns$ and $13.7ns$, respectively. Therefore, we sort them as $\{a, c, b\}$. We

INPUT: Routing graph and terminal locations ;
OUTPUT: Steiner point set S, $RSMT = MST\{S \cup T\}$;
Initialization: Steiner point set $S = \Phi$;
Initialization: $l_0 = MST(T)$, $flag = 0$;
Build the circuit model according to the routing graph and the terminal locations;
Simulate the circuit;
Sort the Hanan node into a sequence E according to their peak time;
WHILE {There are less than $n-2$ nodes in S and $E \neq \Phi$ and $flag < n/8$}
  WHILE the 1st node A in E is in the current tree
    Remove A from E;
  END
  Select the first node A in E;
  $l_1 = MST\{S \cup T \cup A\}$;
  IF $\{l_1 < l_0\}$
    $S = S \cup A$;
    $l_0 = l_1$;
    $flag = 0$;
  ELSE
    $flag = flag + 1$;
  END
  Remove A from E;
END

Fig. 8. 1-cktSteiner Algorithm.

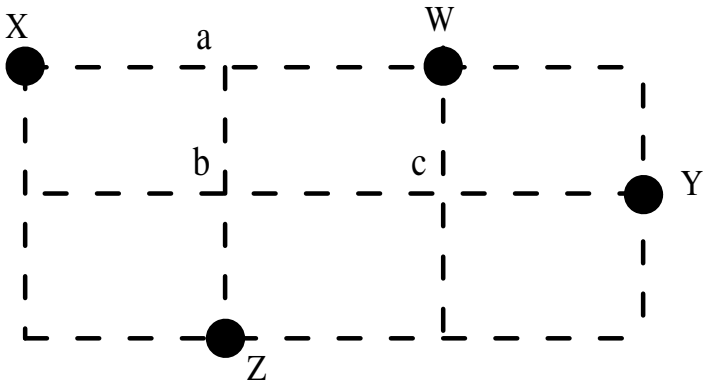

Fig. 9. A four-terminal testcase to illustrate the steps for 1-cktSteiner algorithm.

then add a and construct the MST on node set$\{W, X, Y, Z, a\}$. The total wirelength for the tree is 6, which is smaller than 7, the wirelength of the MST for the node set $\{W, X, Y, Z\}$. So we keep node a as the Steiner point. Then we construct MST on node set $\{W, X, Y, Z, a, c\}$ and the wirelength is still 6. So we discard c, and compute the wirelength of the MST on the node set $\{W, X, Y, Z, a, b\}$, which is still 6. Therefore, we return with Steiner point a, and the optimal wirelength 6.

In general, more than one Steiner node can be added each time for the algorithm in Figure 8. We call the nodes to be added simultaneously as a *block* of nodes. If the total wirelength using a block is reduced, then we take all the nodes in the block as Steiner points; otherwise, we check the vertices in the block one-by-one. In

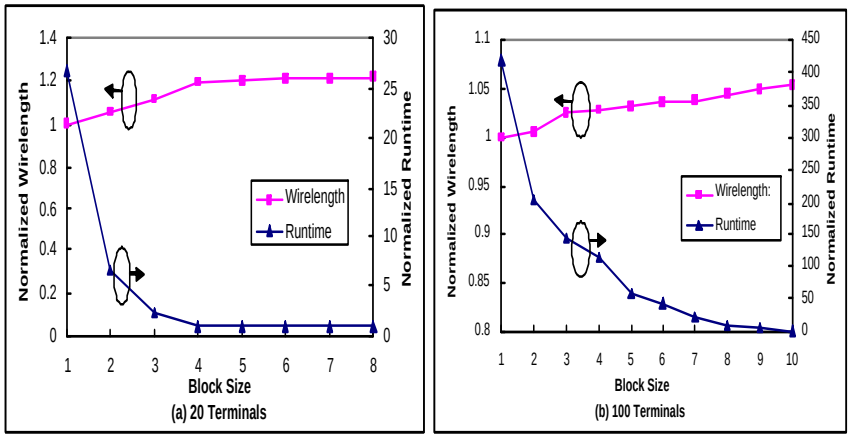

Fig. 10. Tradeoff between wirelength and runtime in (a) 20 terminal case and (b) 100 terminal case. The wirelength and runtime are normalized.

this case, *1-cktSteiner* algorithm becomes a block based algorithm, and the number of points in a block is called *block size*. In Figure 10, 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* algorithm.

3.2 Practical Implementation Issues

It might take a long time to simulate the circuits, especially when the net has a large number of terminals. To reduce circuit simulation time, we present a lookup table approach. The table is built for each routing plane with given routing grid and can be used for *any* net in this routing plane.

To build the table, we compute the output waveforms at all nodes with an impulse current source at only one node. This results in N^3 waveforms for an $N \times N$ routing grid. All those waveforms are stored as a table. To more efficiently simulate the circuits, techniques such as model order reduction [Odabasioglu et al. 1998]and random walk [Qian et al. 2003] can be applied.

When the table is set up, we can directly get the simulation results for any net by superposition, i.e., we sum the waveforms generated by the current source at each terminal to obtain the waveform. Take a three-terminal case for example. The terminals are located at node #1, #6 and #10. We have pre-computed the waveform for the case when there is an impulse current source at node #1 as well as the waveforms when the single current source is added at node #6(#10). By superposing those three waveforms, we can get the final waveform, which is equivalent to imposing three current sources simultaneously at the three terminals as required by our algorithm. Then we look for the peak time for the superposed

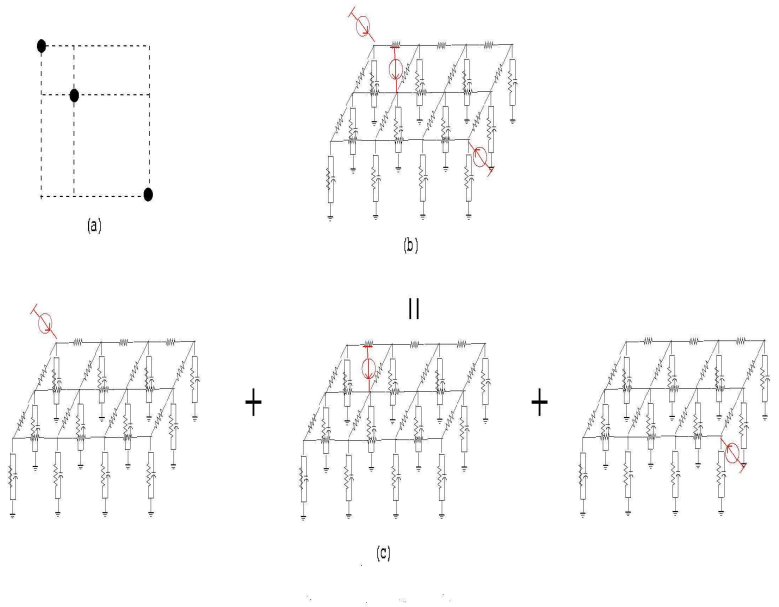

Fig. 11. Compute the waveforms for the circuit model corresponding to a 3-terminal net by table look-up.

waveform. An example of such waveform superposition is already shown in Figure 6. The procedure is illustrated in Figure 11. As discussed in Footnote 1, the circuit model and therefore this table-based waveform calculation is independent of obstacles.

Finally, numerical error and instability usually prevent us from accurately finding the peak time for a particular waveform. Therefore, in practical implementation, rather than seeking for the exact peak time, we seek at the rising edge for the time where the voltage response reaches αV_{max} $(0 < \alpha < 1)$, which allows us more flexibility.

4. EXPERIMENTS AND DISCUSSIONS

We implement the circuit construction and simulation in MATLAB and the tree construction part in C. We run experiments on a few groups of testcases, each group for a selected number of terminals. We generate twenty testcases for each group, with terminals randomly placed in a routing plane with 1000×1000 grid. The circuit models (RC mesh of different granularities) are pre-built before routing for different nets, as have been discussed in Section 3.2. 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. In addition, we compare our algrotihm with a commercial tool Talus PX [mag] and test them on real industrial circuits. The wirelength and runtime are reported as well. For all the experiments below, we always use $B = \frac{n-2}{8}$ for *B-cktSteiner*.

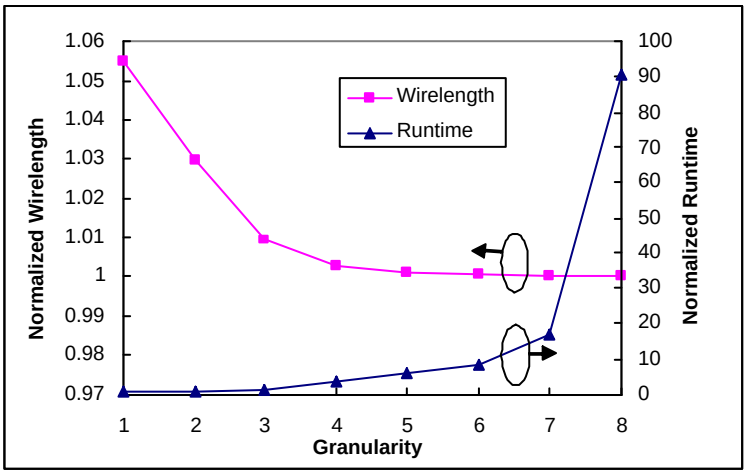

Fig. 12. Normalized runtime and wirelength with respect to RC mesh granularity. The testcase has 20 terminals and is routed by *1-cktSteiner*.

4.1 Impact of Granularity of the RC Mesh

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 12 illustrates how the grid granularity influences wirelength and runtime. We use a testcase 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 increases by $3X$ compared to using the original grid (equivalent to GRG). A similar tradeoff has been observed for other testcases as well. Therefore, we use a 4X finer grid in all experiments below.

4.2 Obstacle-free Routing

Table I presents experiments for obstacle-free routing. We first compare *1-cktSteiner* with the exact solution of *GeoSteiner* [Warme and et al], and *FastSteiner* which generates the shortest wirelength among existing heuristics. Both *1-cktSteiner* and *FastSteiner* [Kahng et al. 2003] 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 I also compares *B-cktSteiner* with *FLUTE* [Chu and Wong 2005], the fastest algorithm among existing heuristics. *B-cktSteiner* reduces up to 1.9% wirelength with a similar runtime when compared to *FLUTE*. On average, *B-cktSteiner* is 3.4% worse than the exact solution in terms of wirelength, 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. The runtime of *B-cktSteiner* is $24X$ smaller.

We also compare our algorithm with the Magma routing package [mag] on real industrial circuits. Both packages are implemented inside the Magma tool flow,

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

ckt	# of nets	pin #			# of nets		wirelength imp		total runtime ratio
		min	max	aver	gained	tied	max	total	ckt/[1]
#1	990	2	16	12	180 (18.2%)	550 (55.6%)	3.20%	-0.11%	0.99
#2	1840	2	49	16	610 (33.1%)	960 (52.2%)	3.29%	0.25%	0.99
#3	2400	2	20	10	1000 (41.7%)	1200 (50.0%)	6.25%	1.12%	0.99
#4	1500	2	811	135	400 (26.7%)	1100 (73.3%)	0.53%	0.12%	1.23
#5	5190	2	30	14	990 (19.1%)	3390 (65.3%)	3.39%	0.01%	1.00
#6	7900	2	21	13	1300 (16.5%)	6000 (76.0%)	6.67%	1.11%	0.99
#7	6320	2	58	19	1930 (30.5%)	2620 (41.5%)	8.15%	1.02%	1.31
#8	5720	2	39	16	2060 (36.0%)	2240 (39.2%)	0.22%	0.05%	1.05
#9	8600	2	30	19	2322 (27.0%)	4610 (53.6%)	2.44%	0.74%	1.00
#10	3420	2	1927	30	1010 (29.5%)	2200 (64.3%)	3.55%	1.65%	1.21
Aver	4388	2	300	28	1180 (26.9%)	2487 (56.7%)	3.77%	0.60%	1.07

Table II. Chip-level comparison between 1-cktSteiner and an industrial routing package.

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

and the experiments are run on AMD Opteron double-CPU servers (2.6G CPUs, 4-8G RAM). The results are reported in Table II. As we can see from the table, compared with Magma routing package containing eight routing algorithms, 1-cktSteiner reduces the chip level total wirelength by up to 1.23% and reduces net-based wirelength by up to 8.15%.

4.3 Obstacle-avoiding Routing

Table III 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 testcases at a similar runtime, and *B-cktSteiner* has an average speedup of $352X$ with wirelength similar to that produced by *An-OARSMan*. Because the global router in Magma tool Talux PX is not able to consider obstacles, we are not able to compare in an industrial setting.

4.4 The Number of Steiner Points Required for the Optimal Solution

We also illustrate 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 testcases for both obstacle-free and obstacle-avoiding

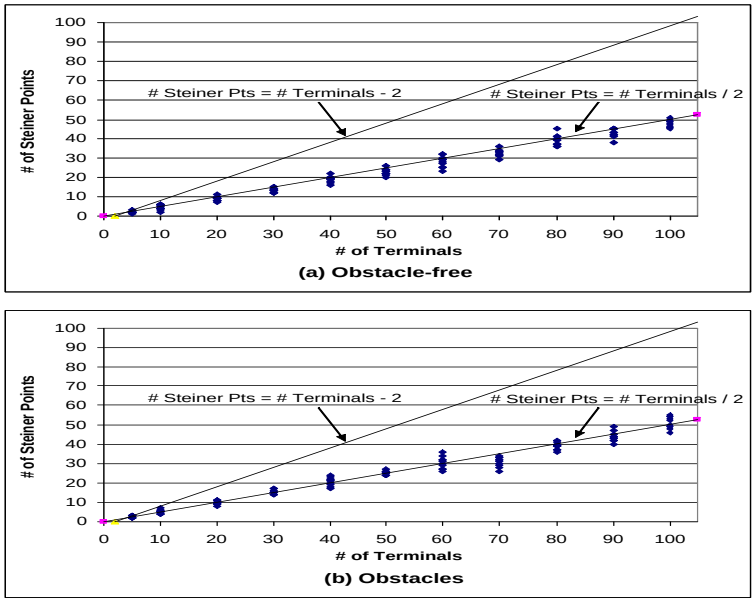

Fig. 13. The number of terminals n versus the number of Steiner points q used by *1-cktSteiner*: (a) obstacle-free routing and (b) obstacle-avoiding routing

routings. The result is shown in Figure 13. 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 may be used to develop more efficient algorithms in the future.

5. CONCLUSIONS AND DISCUSSIONS

Using an RC network to simulate routing, we show in this paper that Steiner points can be selected based on circuit behavior. Then, any routing algorithm can apply the selected Steiner points to construct (rectilinear) Steiner minimum tree (RSMT). In this paper, we apply selected Steiner points to *1-Steine*r algorithms and develop 1-cktSteiner algorithm and a faster version *b-cktSteiner* algorithm. When constructing RSMT without obstacles, *1-cktSteiner* obtains similar length but 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 similar runtimes 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.

The package of *cktSteiner* can be downloaded at http://eda.ee.ucla.edu/tools.html. In the future, we will extend circuit simulated routing to routing congestion estimation, and routing for performance and other routing objectives and constraints.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Hsiao-Ping Tseng and Dr. Yangdong Deng at Magma Design Automation for help on chip-level comparison with Magma router. The authors would also like to thank the anonymous reviewers for their review comments, which helped to improve the presentation of this work significantly.

REFERENCES

http : //www.magma − da.com/c/@ei__ci__2hsu.41ji/pages/productsintro.html.

Akers, S. B. 1967. A modification of lee's path connection algorithm. *IEEE Trans. on Electronic Computer 16,* 4, 97–98.

Albrecht, C. 2000. Provably good global routing by a new approximation algorithm for multi-commodity flow. In *Proceedings of ISPD.*

Atkinson, D. and van Steenwijk, F. J. 1999. Infinite resistive lattices. *Am. J. Phys.*, 486–492.

Chu, C. and Wong, Y.-C. 2005. Fast and accurate rectilinear steiner minimal tree algorithm for vlsi design. In *Proceedings of ISPD.*

Ganley, J. L. and Cohoon, J. P. 1994. Routing a multi-terminal critical net: Steiner tree construction in the presence of obstacles. In *Proceedings of ISCAS.*

Garey, M. R. and Johnson, D. S. 1977. The rectilinear steiner tree problem is np-complele. *SIAM Journal on Applied Mathematics*, 826–834.

Hadlock. 1977. A shortest path algorithm for grid graphs. *Networks*, 323–334.

Hightower, D. W. 1969. A solution to the line routing problem on the continous plane. In *Proceedings of DAC.*

Hu, Y., Jing, T., Hong, X., Feng, Z., Hu, X., and Yan, G. 2005. An-oarsman: Obstacle-avoiding routing tree construction with good length performance. In *Proceedings of ASPDAC.*

Hwang, F. K., Richards, D. S., and Winter, P. 1992. *The Steiner Tree Problem.* volume 53 of Annals of Discrete Mathematcis, North-Holland, Amsterdam, Netherlands.

Kahng, A. B., I.I.Mandoiu, and Zelikovsky, A. 2003. Highly scalable algorithms for rectilinear and octilinear steiner trees. In *Proceedings of ASPDAC.*

Kahng, A. B. and Robins, G. 1992. A new class of iterative steiner tree heuristics with good performance. *IEEE Trans. on CAD 11,* 7, 893–902.

Kahng, A. B. and Robins, G. 1995. *On Optimal Interconnections for VLSI.* Kluwer Academic Publishers, Boston.

Mandoiuand, I. I., Vazirani, V. V., and Ganley, J. L. 2000. A new heuristic for rectilinear steiner trees. *IEEE Trans. on CAD.*

Mikami, K. and Tabuchi, K. 1968. A computer program for optimal routing printed circuit connectors. In *Proceedings of IFIPS.*

Odabasioglu, A., Celik, M., and Pileggi, L. 1998. PRIMA: Passive reduced-order interconnect macro-modeling algorithm. *IEEE Trans. on CAD*, 645–654.

Qian, H., Nassif, S. R., and Sapatnekar, S. S. 2003. Random walks in a supply network. *Proceedings of DAC.*

Rubin, F. 1974. The lee connection algorithm. *IEEE Trans. on Computer*, 907–914.

Shi, Y., Mesa, P., Yu, H., and He, L. 2006. Circuit simulation based obstacle-aware steiner routing. *Proceedings of DAC.*

Soukup, J. 1978. Fast maze router. In *Proceedings of DAC.*

Warme, D. W. and et al. Geosteiner 3.1. package.

Zachariasen, M. and Winter, P. 1999. Obstacle-avoiding euclidean steiner trees in the plane: an exact algorithm. In *extended abstract presented at the Workshop on Algorithm Engineering and Experimentation.*

Zelikovsky, A. 1993. An 11/6-approximation for the network steiner tree problem. *Algorithmica*, 463–470.

Zheng, S. Q., Lim, J. S., and Iyengar, S. S. 1996. Finding obstacle-avoiding shortest paths using implicit connection graphs. *IEEE Trans. on CAD*, 103–110.

Zhou, H. 2003. Efficient steiner tree construction based on spanning graphs. In *Proceedings of ISPD*.

Received Month Year; Revised Month Year; accepted Month Year