A Negative Reinforcement Method for PGA Routing

F. D. Lewis and Wang Chia-Chi Pong

Department of Computer Science
University of Kentucky
Lexington, Kentucky 40506

***Abstract*—We present an efficient and effective method for the detailed routing of symmetrical or sea-of-gates FPGA architectures. Instead of breaking the problem into 2-terminal net collections we propose a model in which Steiner trees spanning nets of logic blocks are constructed on grids induced by the blocks. Then a new routing technique, *negative reinforcement* is employed to prevent nets from blocking each other. The experimental results are very promising.**

I. Introduction

Programmable Gate Arrays (PGA's) have proven a very attractive paradigm for semi-custom VLSI design, due primarily to the short design and fabrication times required, especially when compared to full custom design methods for VLSI circuits. Recently, the use of Field Programmable Gate Arrays (FPGA's) has emerged as a means of implementing logic circuits for Application-Specific Integrated Circuits (ASIC's) when quick turnaround, reconfigurability, and low-cost prototypes become essential .

The CAD process for FPGA's proceeds through specification, logic optimization, technology mapping, placement, and routing. A major problem in routing occurs when nets overlap and block each other. Solutions to this problem often involve specifying connections as 2-terminal nets and employing maze-routers [12] with rip-up and reroute techniques [1,27,24,5,18,15,26]. Wire-pushing [25] has proven interesting as has introducing a global routing phase [1,27,17]. Special techniques for Symmetrical Array FPGA [4,22,23] have also been examined.

Multiterminal, multinet wiring problems have been examined in general via branch and bound [28], linear programming relaxation [24], traditional rip-up methods [13] and special variations of Steiner trees [6]. There is even a language to describe collections of trees for routing [16].

We turn away from 2-terminal nets and propose a model for detailed routing in which Steiner spanning trees are constructed on a grid induced by the FPGA logic blocks. Overlaps between nets are removed in an iterative fashion with a new *negative reinforcement* technique. This provides a framework in which entire nets may be routed simultaneously thus leading to shorter overall wire length.

II. Preliminaries

A *shortest rectilinear Steiner spanning tree* (or Steiner tree) over a set of points (in the plane) is a minimal length collection of vertical and horizontal lines connecting the points. Hanan [8] showed that one of these exists on the smallest complete rectangular grid containing the points, known as the *grid induced by the points*. Figure 1a contains four points with their induced grid and figure 1c depicts a shortest rectilinear Steiner spanning tree over these points.

Rectilinear minimum spanning trees (MST's) also exist on the induced grid. One is in figure 1b. Hwang [9] assures us that they are no larger than one and a half times the size of the shortest rectilinear Steiner tree spanning the points. And, of course, no spanning tree can be smaller than one half the perimeter of the induced grid.

Heuristics for constructing Steiner trees are often necessary since the problem is NP-complete [7] for more than three points. A survey of these methods appears in [10].

A *grid* is a rectilinear collection of edges and an *edge* runs between *grid intersections*. The induced grids mentioned above are *complete* since inside the perimeter all edges are included. Most of the grids we use below are not complete.

A *tour* of a set of points is a connected sequence of edges on the induced grid, which, when traversed, takes one through all of them. It is a special type of *path*.

III. FPGA Architectures

Several companies provide various types of FPGA's [3]. Two commercially available FPGA families of interest to us are the Symmetrical Array or Island-Type FPGA (from Xilinx and QuickLogic) and the Sea-of-Gates FPGA (from Plessey, Algotronix, and Concurrent).

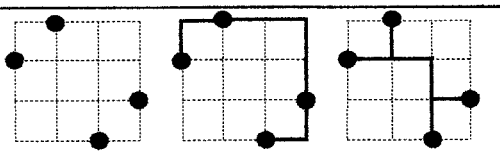


Fig. 1. Induced Grid and Spanning Trees

30th ACM/IEEE Design Automation Conference®

Each is an array of Configurable Logic Blocks (CLB's). In the Symmetrical Array architecture they connect to a grid containing switchboxes. The CLB's in a Sea-of-Gates FPGA connect locally to their neighbors and connections through CLB's (with multiplexers), and over the top of the array (with a dense interconnect resource) may also be made.

IV. The Routing Model

To formulate a model which applies to the architectures featured in the last section, we examine the general routing problem. Our routing area is depicted as a grid with the groups of CLB's to be connected as points at intersections on the grid. Groups of CLB's to be connected are called *nets.* Two nets and their induced grid appear in figure 2a.

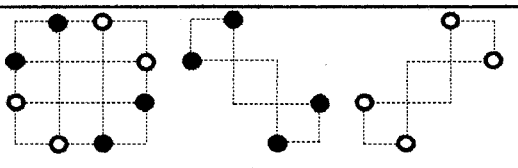
Steiner trees (one connecting each net) on the grid induced by the union of the points in the nets provide our routing. Physical wiring constraints determine exactly which edges of the grid may be used to build these Steiner trees. First, we do not allow parts of a tree spanning a net to cross any of the points (CLB's) from other nets. Thus each tree is *restricted* to a portion of the induced grid. Figure 2b shows the *grid restricted to the black net.* Figure 2c displays that for the white net.

For pairs of nets the notation we shall use is defined as follows. If G is the rectangular grid induced by the nets containing the sets of points P_a and P_b, then the two restricted grids are:

$$G_a = G - \{\text{edges connected to } p_i \in P_b\}$$

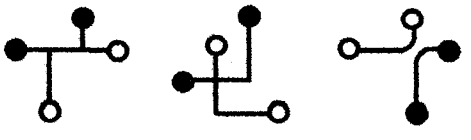
$$G_b = G - \{\text{edges connected to } p_j \in P_a\}$$

Other conventions and constraints apply to our routing model. We shall allow lines in two trees to *cross* at a grid intersection (since we have at least 2-layer routing) or to *turn* at an intersection (this is knock-knee routing) as pictured in (b) and (c) of figure 3. We do not allow *overlaps* as shown in 3a, though this constraint may be relaxed if edges on the routing grid are allowed to have *capacities* greater than one.



(a) (b) (c)

Fig. 2. Nets and Grids Restricted to Nets



(a) (b) (c)

Fig. 3. Routing Conventions

Definition

The *nonoverlapping multiple rectilinear Steiner spanning problem* for two nets containing the sets of points P_a, and P_b is the construction of the smallest pair of rectilinear Steiner spanning trees, $T_a \subseteq G_a$ which spans P_a, and $T_b \subseteq G_b$ which spans P_b such that $T_a \cap T_b = \emptyset$. □

This is the *optimum solution,* the smallest Steiner forest which spans the points in the nets and contains no overlaps. Since at times there may be no nonoverlapping solution, we shall also be interested in minimizing overlaps.

V. Properties of the Problem and the Model

Before attempting to solve this multinet routing problem we examine our model. The following ensures that we will be restricted to heuristic algorithms.

Theorem 1

The nonoverlapping and minimum overlap multiple rectilinear Steiner spanning problems are NP-complete. □

A question of primary importance is whether there is any solution to a particular problem. There are cases for which there is no possible way to route the nets on their restricted grids without overlaps. Consider the two nets in figure 2a with their restricted grids of figures 2b and 2c. There is no way to route both at the same time on these grids without overlaps because each must traverse the central square of their restricted grids. Thus, in several simple cases it is obvious that there is no solution to the multinet rectilinear Steiner spanning problem. Two are:

- A connected component of the restricted grid does not contain all of the points in a net.
- Two restricted grids share an *articulation area* (such as the center square in both restricted grids of figure 2) which must be traversed by each.

and a general symptom for no solution follows.

Theorem 2

There is no solution for the nonoverlapping multiple rectilinear Steiner spanning problem (over two nets) if and only if every pair of net-tours (one on each set of points) on the restricted grids contains an overlap. □

This indicates that determining if there is a solution to the nonoverlapping Steiner problem might be NP-complete as well. For this reason, we develop algorithms to solve the *minimum overlap* problem. Here we easily know when there is a solution.

Theorem 3

There is a solution for the multiple rectilinear Steiner spanning problem if and only if each set of points is in one connected component of its own restricted grid. □

In the single tree case we were able to relate solutions to the size of minimum spanning trees using Hwang's theorem [9] and guarantee an size range. Here we are not so fortunate. Consider the problem depicted in figure 4. The minimum length Steiner spanning forest for the nets is in figure 4a, while figures 4b and 4c show minimum spanning trees over the nets on their restricted grids. This confirms that the sum of the sizes of the minimal spanning trees *is not* an upper bound on the length of the Steiner spanning forest for the nonoverlapping Steiner spanning problem.

We do have two lower bounds. One is merely half the sum of the perimeters of the smallest rectangles enclosing the nets and the other is the sum of the smallest Steiner trees built without regard to overlaps. This means that we still have Hwang's lower bound of two-thirds the minimal spanning tree size if these spanning trees are constructed on the induced rather than the restricted grids.

VI. Single Steiner Trees

Before routing several nets at once, we develop a method for routing one net by generating Steiner trees on the restricted grids for nets. We elect to use an algorithm based upon Kruskal's greedy algorithm for minimal spanning trees [11]. This will be similar in spirit to other Kruskal-based procedures [2, 14] which have been formulated for use on complete induced grids rather than restricted grids.

This means that we must route in the *presence of obstacles* [21, 29] which makes distance computations more difficult since we must search for paths between points rather than computing distances directly. Also, differences in size between minimum spanning trees and Steiner trees are less dramatic than if there were no obstacles.

Following Kruskal, we combine Steiner trees for subnets a greedy manner. We begin with each point as a tree and then connect the pair of closest trees to form a new tree. Then we determine the distance from the edges of this new tree to the remaining trees. This continues until only one tree remains.

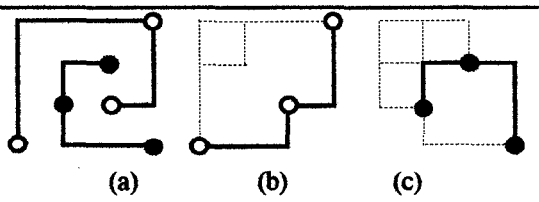


Fig. 4. Steiner Forest and MST's

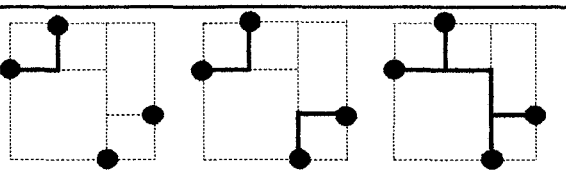


Fig. 5. Building a Steiner Tree

This is illustrated in figure 5 where first the upper left corner points are connected, then the bottom right ones, and at last, both trees are joined.

Since we are building a tree on a grid which is not complete there can be no guaranteed relation between the best and worst-case sizes of the tree.

A note on the complexity is in order. Since we are not routing on a complete grid, we must detect paths rather than calculate distances. Using wavefront propagation in the spirit of the maze crawling algorithm of Lee [12] we might search the entire grid. This means, in the worst case, $O(n^2)$ time for updating the distance matrix used to predict the closest trees and recording the paths between trees. Thus our algorithm has worst case time complexity $O(n^3)$.

Details for all aspects of this algorithm are found in [19].

VII. Simultaneous Tree Generation

We begin by developing an algorithm to route two nets of points at the same time and then extend it to arbitrarily large numbers of nets. Our strategy for routing the two-net case is to first construct Steiner spanning trees on the restricted grids for each net. Next, we check to see if they overlap. If so, we penalize the use of the overlapped edge by pretending that its length has doubled and reconstructing one of the trees. This continues until there are no overlaps, or, until we reach a limit (named maxtries) placed on the number of attempts to correct overlaps. If this limit is reached, the pair of trees previously constructed with the minimum amount of overlap is selected.

Consider figure 6. In 6a both trees overlap on the two grid edges in the shaded area. The grid edges comprising the overlap are assigned larger weights (their length is doubled) and the tree spanning the white points is redrawn in 6b. Since there is still an overlap, the weights of the edge in the shaded area is increased and the tree spanning the black points is redrawn in 6c. At last there is no overlap.

Trees are of course constructed using the Kruskal-based greedy algorithm and checking overlaps between trees is accomplished by sorting the grid edges which make up the trees and comparing them. Since a bucket sort may be used, the complexity is the same as the number of edges in the trees. (And even this can be reduced if we omit the edges connected to points since they're not in the other restricted grid.) Thus the complexity of algorithm is dominated by the time used to build Steiner spanning trees.

Our *negative reinforcement* routing algorithm for pairs of nets is presented in figure 7.

Extending the algorithm to groups of k nets involves iteration over the nets. Consider the code fragment in figure 8. In this algorithm the last tree constructed is rebuilt if overlaps still exist and that if the limit on iterations has been reached a tree is still constructed.

As with the two net case, constructing Steiner trees dominates the complexity of the algorithm. Since up to k trees can be built each time we repeat the inner loop, our complexity is $O(k^2n^3)$.

Another method is to construct the tree at the beginning of the loop, check for overlaps involving *all* trees, and continue on to the next tree. This variation was not found to be significantly different from the algorithm in figure 8.

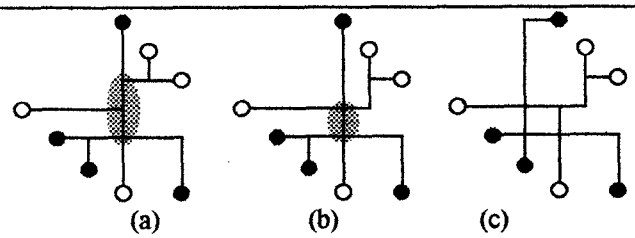


Fig. 6. Routing Two 4-Point Nets

Route(P_a, P_b, G_a, G_b, maxtries, T_a, T_b)
PRE: G_a and G_b = induced grids restricted to P_a and P_b
POST: T_a and T_b = Steiner Trees spanning P_a & P_b on G_a & G_b

```
tries = 0
Construct Steiner tree Ta spanning Pa on Ga
repeat
    Construct Steiner tree Tb spanning Pb on Gb
    CheckOverlaps(Ta, Tb)
        if (overlaps exist) then
            Adjust length of overlapped segments in Ga
        Construct Steiner tree Ta spanning Pa on Ga
            CheckOverlaps(Ta, Tb)
        if (overlapped segments exist) then
                Adjust length of overlapped segments in Gb
            tries ++
until (no overlaps remain) or (tries > maxtries);
```

Fig. 7. Simultaneous Routing Procedure

```
for i = 1 to k do
  CheckOverlaps(T1, ..., Ti-1)
  tries = 0
  while (overlaps exist) and (tries ≤ maxtries) do
      Adjust length of overlapped segments in Gi-1
      Construct Steiner tree Ti-1 spanning Pi-1 on Gi-1
      CheckOverlaps(T1, ..., Ti-1)
      if (overlapped segments exist) then
          for every Tj (1 ≤ j < i-1) with overlapped edges do
              Adjust length of overlapped segments in Gj
              Construct Steiner tree Tj spanning Pj on Gj
          CheckOverlaps(T1, ..., Ti-1)
      tries++
      Adjust length of overlapped segments in Gi
  Construct Steiner tree Ti spanning Pi on Gi
```

Fig. 8. Multinet Routing Algorithm

VIII. Empirical Results

The algorithms were implemented in C and run on a Sequent Symmetry S81 Dynix 3.0 machine with randomly generated data sets. Three parameters were varied:

- number of nets in each data set,
- number of points (n) in each net, and
- size of grid induced by points in the nets.

The latter is necessary since we wish points in different nets to share coordinates. If they do not, then there are shortest Steiner trees spanning each net over the grids *they themselves* induce due to a result of Hanan [8].

Three induced grid sizes (1.25n by 1.25n, 1.5n by 1.5n, and 1.75n by 1.75n) were utilized for data sets. This insured that 75%, 50%, and 25% of the points in each net shared coordinates with the other net.

The negative reinforcement algorithm was run on data sets composed of pairs of 5 to 50 point nets. Each data set contained 10 pairs of nets. For each net, the sum of the sizes of the Steiner spanning trees was compared to the combined sizes of minimal spanning trees for the nets.

Table I compares minimal spanning trees to the trees constructed by the negative reinforcement algorithm for pairs of n point nets on the three induced grid sizes. In the columns labeled 'First' we find the average percent improvement of the initial Steiner trees over minimal spanning trees on the restricted grids, while the 'Last' columns show the average percent improvement after iterating to remove overlaps between the pairs of nets.

As expected, denser grids provided cases for which less improvement was possible. But, non overlapping Steiner spanning trees were found for every data set. Most data sets required no more than three iterations to remove overlaps, and the worst data set (1.25n grid with 5 points) needed six iterations.

Table II and table III provide the average improvement for three and five net data sets on the same size induced grids.

These data sets were packed even more densely than the pairs of nets, and so provided even less pleasing results. In fact, we could generate no data sets on the 1.25n grid for the five-net case because articulation areas always appeared.

TABLE I
Percent Improvement for 2-Net Data Sets

	1.75n Grid		1.5n Grid		1.25n Grid	
n	First	Last	First	Last	First	Last
5	7.05	6.95	6.55	3.23	4.86	1.17
10	6.99	6.86	6.65	6.07	6.82	5.15
20	5.59	5.59	5.61	5.33	5.62	4.79
30	6.06	5.89	6.58	6.24	5.10	5.00
50	5.57	5.54	5.46	5.37	5.33	5.06
Ave	6.25	6.17	6.17	5.25	5.55	4.23

TABLE II
PERCENT IMPROVEMENT FOR 3-NET DATA SETS

n	1.75n Grid First	1.75n Grid Last	1.50n Grid First	1.50n Grid Last	1.25n Grid First	1.25n Grid Last
5	6.60	6.07	4.45	3.37	6.10	0.61
10	6.63	6.51	5.99	5.07	5.16	3.78
15	5.23	4.88	5.66	4.94	5.45	3.94
20	5.61	5.42	5.14	4.65	6.05	4.65
Ave	6.02	5.72	5.31	4.51	5.69	3.24

TABLE III
PERCENT IMPROVEMENT FOR 5-NET DATA SETS

n	1.75n Grid First	1.75n Grid Last	1.50n Grid First	1.50n Grid Last	1.25n Grid First	1.25n Grid Last
5	6.48	4.33	4.25	-1.77	2.18	
10	6.17	5.89	5.78	4.82	6.02	2.41
15	5.56	5.32	4.71	3.33	5.70	3.01
20	4.84	4.70	5.61	5.06	5.53	1.14
Ave	5.76	5.06	5.09	2.86	4.86	2.19

Reduction of overlaps was very successful. In all of the test cases, *only eight resulted in spanning tree forests which contained overlaps!* The number of iterations in the three-net case was less than usually less than five with one at seven, and the five-net data sets normally required no more than 10 iterations with a maximum of 18 iterations.

IX. CONCLUSION

We have presented an efficient and effective method for the detailed routing of symmetrical or sea-of-gates FPGA architectures. Part of this is due to the fact that it is based upon simultaneous Steiner spanning trees rather than sequential routing of 2-terminal net collections. In the proposed model, Steiner trees spanning nets of logic blocks are constructed on grids induced by these blocks by an algorithm guaranteed to be at least as good as an MST algorithm applied to the modified grids.

The new routing technique, negative reinforcement, is employed to prevent nets from blocking each other. Experiments have been successful in providing nonoverlapping routings in 97% of the test cases.

REFERENCES

[1] Adams, G. D., and Séquin, C. H. "Template Style Considerations for Sea-of-Gates Layout Generation." *Proc. of 26th ACM/IEEE DAC*, 1989, 31-36.

[2] Bern, M. W. and deCarvalho, V. "A Greedy Heuristic for the Rectilinear Steiner Tree Problem." *Tech. Rpt. Comp. Sci. Div.*, UC Berkeley,1985.

[3] Brown, S. D., Francis, R. J., Rose, J. and Vranesic, Z. G. *Field-Programmable Gate Arrays*, Kluwer Academic Publishers, Boston, 1992.

[4] Brown, S. D., Rose, J., and Vranesic, Z. G. "A Detailed Router for Field-Programmable Gate Arrays." *Proc. IEEE ICCAD*, 1990, 382-385.

[5] Chakraverti, A. and Chung, M. J. "Routing Algorithm for Gate Array Macro Cells." *Proc. 25th ACM/IEEE DAC*, 1988, 658-662.

[6] Chiang, C., Sarrafzadeh, M., and Wong, C. K. "A Powerful Global Router Based on Steiner Min-Max Trees." *Proc. IEEE ICCAD, 1989, 2-5.*

[7] Garey, M. R. and Johnson, D. S. "The Rectilinear Steiner Problem is NP-Complete." *SIAM J. Appl. Math.*, vol. 32, 1977, 826-834.

[8] Hanan, M. "On Steiner's Problem with Rectilinear Distance." *SIAM J. Appl. Math.*, vol. 14, no. 2, 1966, 255-265.

[9] Hwang, F. K. "On Steiner Minimal Trees with Rectilinear Distance." *SIAM J. Appl. Math.*, vol. 30, 1976, 104-114.

[10] Hwang, F. K., Richards, D. S., and Winter, P. *The Steiner Tree Problem. Annals of Discrete Mathematics*, no. 53, North Holland, 1992.

[11] Kruskal, J. B. "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem." *Proc. of the AMS*, vol. 7, 1956, 48-50.

[12] Lee, C. Y. "An Algorithm for Path Connections and Its Applications." *IRE Trans. on Electronic Computers*, vol. EC-10, Sept 1961, 346-365.

[13] Lee, J. H., Bose, N. K., and Hwang, F. K. "Use of Steiner's Problem in Suboptimal Routing in Rectilinear Metric." *IEEE Trans. on Circuits Syst.*, vol. CAS-23, 1976, 470-476.

[14] Lewis, F. D. and Van Cleave, N. "Correct and Provably Efficient Methods for Rectilinear Steiner Spanning Tree Generation," *Proceedings of the First Great Lakes Computer Science Conference*, and *Springer-Verlag Lecture Notes*, vol. 507, 1991.

[15] Nair, R. "A Simple Yet Effective Technique for Global Wiring." *IEEE Trans. on CAD*, vol. CAD-6, no. 2, March 1987, 165-172.

[16] Ng, A. P-C., Raghavan, P., and Thompson, C. D. "A Language for Describing Rectilinear Steiner Tree Configurations." *Proc. 23rd ACM/IEEE DAC*, 1986, 659-662.

[17] Okuda, R. and Oguri, S. "An Efficient Routing Algorithm for SOG Cell Generation on a Dense Gate-Isolated Layout Style." *Proc. 29th ACM/IEEE DAC*, 1992, 676-681.

[18] Palczewski, M. "Plane Parallel A* Maze Router and its Application to FPGA's." *Proc. 29th ACM/IEEE DAC*, 1992, 691-697.

[19] Pong, W. C.-C. "Steiner Trees and CAD Algorithms for VLSI Systems." *PhD Dissertation*, University of Kentucky, 1993.

[20] Raghavan, P., and Thompson, C. D. "Multiterminal Global Routing: A Deterministic Approximation Scheme." *Algorithmica*, vol. 6, 11991, 73-82.

[21] Rezende, P. J., Lee, D. T., and Wu, Y. F. "Rectilinear Shortest Paths in the Presense of Rectangular Barriers." *Discrete & Comput. Geom.*, vol. 4, 1989, 41-53.

[22] Rose, J. and Brown, S. "The Effect of Switch Box Flexibility on Routability of Field Programmable Gate Arrays." *Proc. 1990 CICC*, 1990, 27.5.1-27.5.4.

[23] Rose, J. and Brown, S. "Flexibility of Interconnection Structures in Field-Programmable Gate Arrays." *IEEE J. of Solid State Circ.*, vol. 26, no. 3, March 1991, 277-282.

[24] Rowson, J. and Trimberger, S. "Gate Array Macro Layout Automation." *Proc. 1986 IEEE ICCAD*, 1986, 448-451.

[25] Shin, H. and Sangiovanni-Vincentelli, A. "A Detailed Router Based on Incremental Routing Modifications: Mighty." *IEEE Trans. on CAD*, vol. CAD-6, no. 6, November 1987, 942-955.

[26] Suzuki, K., Matsunaga, Y., Tachibana, M. and Ohtsuki, T. "A Hardware Maze Router with Application to Interactive Rip-Up and Reroute." *IEEE Trans. on CAD*, vol. CAD-5, no. 4, October 1986, 466-476.

[27] Tzeng, P. and Séquin, C. H. "Codar: A Congestion-Directed General Area Router." *Proc. of 1988 IEEE ICCAD*, 1988, 30-33.

[28] Yang, Y. Y. and Wing, O. "On a Multinet Wiring Problem." *IEEE Trans. Circuit Theory*, vol. CT-20, 1973, 250-252.

[29] Wu, Y. F., Widmayer, P., Schlag, M. D. F., and Wong, C. K. "Rectilinear Shortest Paths and Minimum Spanning Trees in the Presence of Rectilinear Obstacles." *IEEE Trans. on Comput.*, vol. C-36, 1987, 321-331.