SIAM J. COMPUT. © 2006 Society for Industrial and Applied Mathematics
Vol. 35, No. 3, pp. 729–740

THE RECTILINEAR STEINER ARBORESCENCE PROBLEM IS NP-COMPLETE[*]

WEIPING SHI[†] AND CHEN SU[‡]

Abstract. Given a set of points in the first quadrant, a rectilinear Steiner arborescence (RSA) is a directed tree rooted at the origin, containing all points, and composed solely of horizontal and vertical edges oriented from left to right, or from bottom to top. The complexity of finding an RSA with the minimum total edge length for general planar point sets has been a well-known open problem in algorithm design and VLSI routing. In this paper, we prove the problem is NP-complete in the strong sense.

Key words. Steiner tree, computational complexity, NP-complete

AMS subject classifications. 68Q17, 68U05, 90C27

DOI. 10.1137/S0097539704371353

1. Introduction.

Let $P = \{p_1, p_2, \ldots, p_n\}$ be a set of points in the first quadrant of E^2, where $p_i = (x_i, y_i)$. A rectilinear Steiner arborescence (RSA) for P is a directed Steiner tree T rooted at the origin, containing all points in P, and composed solely of horizontal and vertical line segments oriented from left to right, or from bottom to top. A rectilinear Steiner minimum arborescence (RSMA) for P is an RSA for P that has the shortest possible total edge length.

The difference between an RSA and the traditional rectilinear Steiner tree is that an RSA is also a shortest distance tree with respect to the origin. Figure 1.1 shows a Steiner minimum arborescence, a Steiner minimum tree, and a minimum spanning tree for the same set of points with p_1 being the origin. For an introduction on Steiner trees, see the book by Hwang, Richards, and Winter [9].

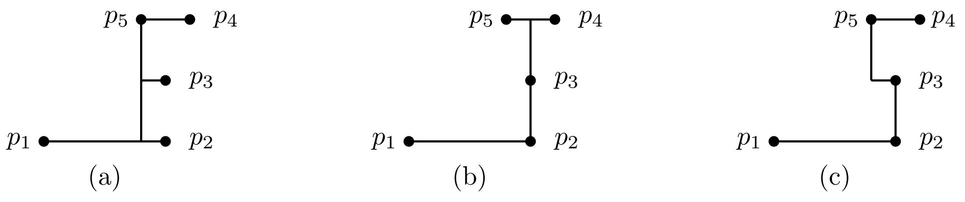

FIG. 1.1. *A rectilinear Steiner arborescence* (a), *a rectilinear Steiner tree* (b), *and a rectilinear spanning tree* (c).

The rectilinear Steiner arborescence problem was first studied by Nastansky, Selkow, and Stewart [13] in 1974. They proposed an integer programming formulation, which has exponential time complexity. In 1979 Laderia de Matos [10] proposed

[*]Received by the editors December 1, 2004; accepted for publication (in revised form) September 29, 2005; published electronically February 3, 2006. This research was supported in part by NSF grants CCR-0098329, CCR-0113668, and EIA-0223785.
http://www.siam.org/journals/sicomp/35-3/37135.html

[†]Department of Electrical Engineering, Texas A&M University, College Station, TX 77843 (wshi@ee.tamu.edu).
[‡]Inet Technologies, Richardson, TX 75081.

an exponential time dynamic programming algorithm. In 1985, Trubin [15] claimed the RSA problem can be solved in polynomial time. In 1992, Rao, Sadayappan, Hwang, and Shor [14] showed that Trubin's algorithm is incorrect and presented an $O(n \log n)$ time approximation algorithm that produces an RSA of length at most 2 times the optimal [14]. In 1994, Córdova and Lee [6] extended the heuristic to points in all four quadrants with the same time complexity. In 1997, Cho [3] again claimed that the RSA problem can be solved in polynomial time using a min-cost max-flow approach. Soon after, Erzin and Kahng showed Cho's claim is wrong [7]. In 2000, Lu and Ruan [12], motivated by the polynomial time approximation scheme (PTAS) of Arora [2], designed a PTAS for the RSA problem. Their PTAS runs in time $O(n^{O(c)} \log n)$ and produces an RSA of length at most $(1 + 1/c)$ times the optimal. However, whether there exists a polynomial time algorithm for the RSA problem remains open.

Because an RSMA is a shortest distance tree of minimum total length, it has important applications in VLSI routing. Cong, Leung, and Zhou [5] showed that routing trees based on RSMAs may have significantly less delay than those based on the traditional Steiner trees. Many researchers proposed efficient heuristics and exponential time exact algorithms for the RSA problem [1, 4].

2. NP-completeness.

2.1. Overall strategy.

We assume that readers have the general knowledge of NP-completeness [8]. The decision version of the RSA problem is as follows:

Instance: A set of points $P = \{p_1, p_2, \ldots, p_n\}$ in the plane, and a positive integer k.

Question: Is there an RSA of total edge length k or less?

The proof is a reduction from planar 3SAT, which was proven to be NP-complete in the strong sense by Lichtenstein [11]:

Instance: A set of variables $V = \{v_1, v_2, \ldots, v_n\}$ and a set of clauses $C = \{c_1, c_2, \ldots, c_m\}$. Each clause contains at most 3 literals. Furthermore, the bipartite graph $G = (V \cup C, E)$ is planar, where $E = \{(v_i, c_j) \mid v_i \in c_j \text{ or } \overline{v_i} \in c_j\}$.

Question: Is there an assignment for the variables so that all clauses are satisfied?

For example, Figure 2.1 is the graph of a planar 3SAT instance $c_1 = v_1 \vee v_2 \vee \overline{v_3}$ and $c_2 = \overline{v_1} \vee \overline{v_4}$. We will use this example throughout the paper.

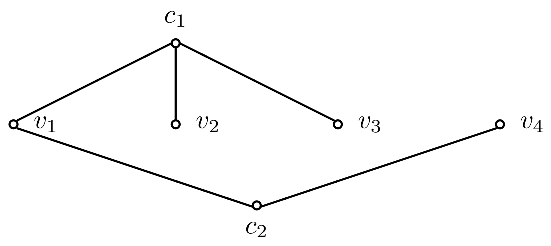

FIG. 2.1. *Graph G of a planar 3SAT instance.*

The reduction is through component design. For a given planar 3SAT instance, we first convert its planar graph G to a planar graph H. Then we embed H in a grid and let the embedded graph be R. Finally we replace each vertex of R by a tile. We will use *basic tiles* to represent variables, *NOT tiles* to negate variables, *OR tiles* to compute the OR of two variables, and *clause tiles* to check if the clauses are satisfied. The length of the RSMA for the set of points so constructed will tell us whether the planar 3SAT has a satisfying solution.

2.2. Embedding. We first convert G of the given planar 3SAT instance to a planar graph H with maximum degree 3.

For each variable v_i in G, let $d(v_i)$ be the degree of v_i. Replace v_i by a path of $d(v_i)$ *variable vertices* for v_i in H: $u_{i1}, \ldots, u_{id(v_i)}$ and edges $(u_{i1}, u_{i2}), \ldots, (u_{id(v_i)-1}, u_{id(v_i)})$. See Figure 2.2 for an example. Each variable vertex will be connected to a clause vertex defined next.

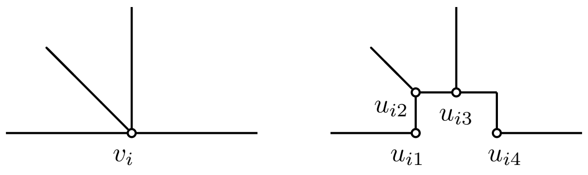

FIG. 2.2. *Vertex v_i of degree 4 is replaced by a path of 4 vertices.*

For each clause c_j in G that contains two variables, if $c_j = v_i \vee \overline{v_k}$, then H will contain a *clause vertex* c_j and two edges (c_j, u_i) and (c_j, u_k), where u_i and u_k are variable vertices for v_i and v_k, respectively. Since the transformation in Figure 2.2 produced $d(v_i)$ variable vertices for each v_i, we make each edge connect to a unique variable vertex. If c_j is not in the right form, say, $c_j = \overline{v_i} \vee \overline{v_k}$, then in H we insert a *NOT vertex* w_{ij} between c_j and the variable vertex for v_i. In other words, we will have a new vertex w_{ij} and edges (c_j, w_{ij}), (w_{ij}, u_i), and (c_j, u_k).

For each clause c_j in G that contains three variables, let $c_j = v_i \vee \overline{v_k} \vee \overline{v_l} = (v_i \vee \overline{v_k}) \vee \overline{v_l} = c'_j \vee \overline{v_l}$. In H, there will be an *OR vertex* c'_j that connects the variable vertices for v_i and v_k, and a *clause vertex* c_j that connects c'_j and the variable vertex for v_l. Similarly, if c_j is not in the right form, we will insert *NOT vertices* as needed.

Now H is a planar graph with maximum degree 3. Figure 2.3 shows the converted planar graph H.

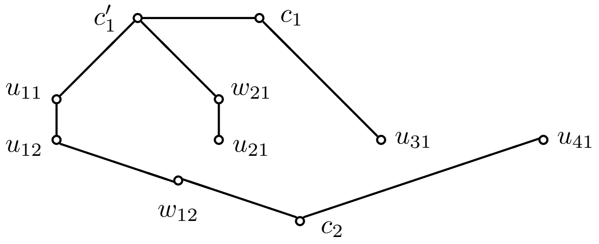

FIG. 2.3. *Planar graph H converted from G.*

Valiant [16] showed that any planar graph $G = (V, E)$ of maximum degree 3 has a planar embedding in a rectilinear grid of area $O(|V|^2)$. Therefore, graph H can be embedded in a rectilinear grid of area of $O((n+m)^2)$, where n and m are the number of variables and clauses. We further require the following in the grid embedding.

(1) Vertices share an edge if and only if they are distance 1 apart.

(2) For each clause vertex $c_j = v_i \vee \overline{v_k}$, or $c_j = c'_j \vee \overline{v_k}$, the path from v_i, or c'_j, enters from the left, and the path from v_k enters from below.

(3) Each OR vertex $c'_j = v_i \vee \overline{v_k}$ occupies two horizontally adjacent vertices in the embedding, the path from v_i enters from the left, the path from v_k enters from below, and the path leading to c_j exits to the right.

These requirements can be satisfied by locally rearranging the grid embedding, increasing the size of the grid by a constant factor, and adding additional vertices and edges. For requirement (2), since v_i, c_j, and v_k is a path with no connection to other vertices in planar graph H, it is always possible to swirl the path to any direction. Figure 2.4 shows the embedding of an example clause vertex $c_j = v_i \vee \overline{v_k}$. For requirement (3), the same idea can be used. For an OR vertex $c'_j = v_i \vee \overline{v_k}$, we first swirl the path to c_j to the right. Then if v_k and v_i are in clockwise order from the path to c_j, then v_k and v_i can be swirled into proper positions. If v_k and v_i are in counterclockwise order from the path to c_j, then a crossing may occur. To avoid the crossing, note that $v_i \vee \overline{v_k} = \overline{v_k} \vee \overline{\overline{v_i}}$. Therefore we can add NOT vertices to paths for v_i and v_k and then route $\overline{v_k}$ to enter from the left and $\overline{v_i}$ to enter from below, thereby maintaining the planarity.

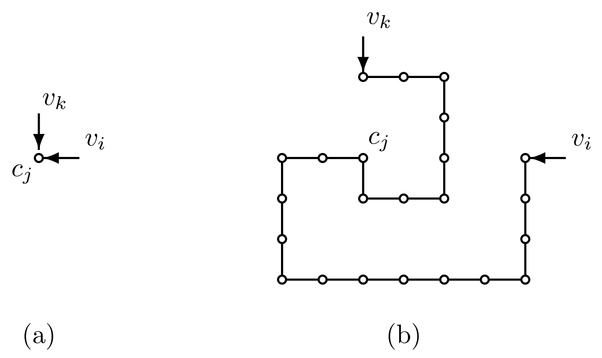

FIG. 2.4. *A clause vertex with incoming paths* (a). *Requirement* (2) *is satisfied by swirling* (b).

Figure 2.5 shows the grid embedding R. There are two *auxiliary vertices* t_1 and t_2 introduced in order to meet the above requirements.

2.3. Component design. The most important building block of our design is the *quadruped* in Figure 2.6. A quadruped consists of 4 *white points* $q_1, \ldots, q_4$, and 4 *black points* $b_1, \ldots, b_4$. The distances between the points are given in Figure 2.6, where $\alpha, \beta, \delta_1, \delta_2 > 0$. In addition, $\beta + \delta_1 > \alpha$, meaning that the Y-coordinate of q_3 is less than the Y-coordinate of q_4, and $\alpha > \delta_1 + \delta_2$, meaning that the rectilinear distance between q_2 and q_4 is $\beta - \delta_2 + \alpha - \delta_1 > \beta$. There is no other point on or inside the region enclosed by the solid, dashed, and dotted lines, i.e., three triangles and one rectangle. We call the region the *forbidden region.*

DEFINITION 2.1. *For a set of points Q that contains white points and black points, a* minimum forest *of Q is a set of RSAs that contains all white points in Q. Furthermore, the root of each RSA is a black point in Q, and the total edge length is minimum.*

LEMMA 2.2. *For a quadruped, there are only two minimum forests, shown as solid edges and dashed edges in Figure* 2.6. *The edge length of both minimum forests is* $2(\alpha + \beta)$.

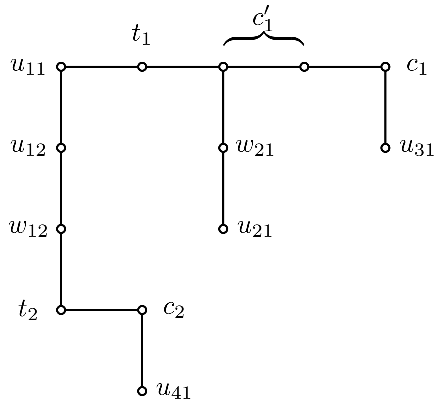

FIG. 2.5. *Embedded graph R.*

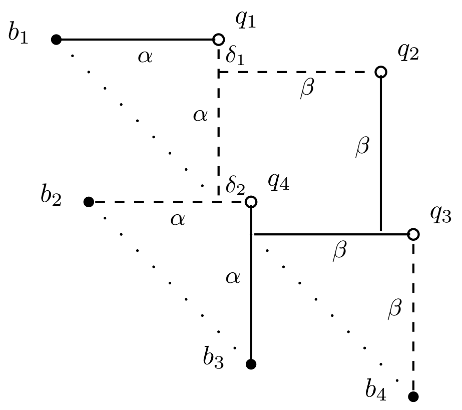

FIG. 2.6. *A quadruped and its forbidden region. The two minimum forests both have total edge length* $2\alpha + 2\beta$.

Proof. To connect q_4, we need a path of length at least α. To connect q_1, we also need a path of length at least α, even with the help of the path for q_4. Similarly, to connect q_2 and q_3, we need two paths each of length at least β. Hence, the lower bound is $2(\alpha + \beta)$.

On the other hand, to connect $q_1, q_2, \dots, q_4$ with the minimum edge length $2(\alpha + \beta)$, each point must use a path that equals the lower bound. Starting with q_2, we can use either the horizontal edge or the vertical edge, both of length β. If we use the vertical edge, then we must connect q_3 with the horizontal edge, q_4 with the vertical edge, and q_1 with the horizontal edge. This gives a total length of $2(\alpha + \beta)$ shown as the solid edges. If we use the horizontal edge for q_2, we will end up with the dashed edges, also of length $2(\alpha + \beta)$. $\square$

We are now ready to define the tiles. The height and width of all tiles, except for the OR tile, are both 96. The height of the OR tile is 96, and the width is 192, twice the width of other tiles.

A basic tile is made of overlapping quadrupeds as shown in Figure 2.7.

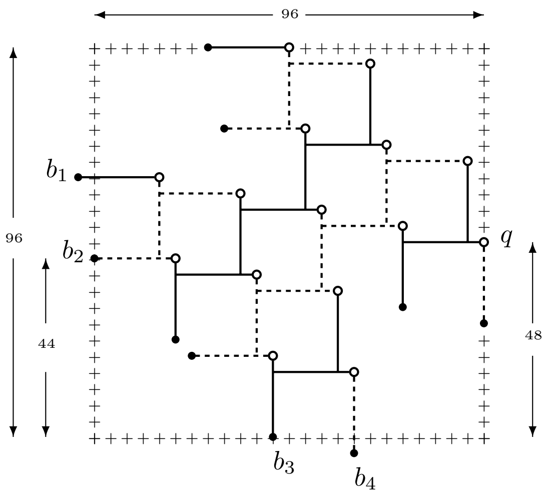

FIG. 2.7. *A basic tile made of overlapping quadrupeds, where $\alpha = \beta = 20$ and $\delta_1 = \delta_2 = 4$.*

LEMMA 2.3. *There are only two minimum forests for a basic tile, one shown as dashed edges and one shown as solid edges.*

Proof. Since the basic tile consists of overlapping quadrupeds, from Lemma 2.2, the minimum forest for one quadruped will force the minimum forests for the rest of the quadrupeds. □

Define the *parity* π of a minimum forest of a basic tile to be 1 if the rightmost white point is connected by a horizontal edge, and 0 otherwise. In Figure 2.7, since the rightmost white point q is connected by a solid horizontal edge, the minimum forest that uses all solid edges has parity 1, and the minimum forest that uses all dashed edges has parity 0.

A set of overlapping quadrupeds such as a basic tile has the important property that once we choose the connection of any point to be solid or dashed, then the entire minimum forest must be solid or dashed. Therefore, if we insist on the connection being a minimum forest, then our choice of any point is propagated left, right, top, and bottom, by overlapping quadrupeds such as basic tiles.

Figure 2.8 shows how to place two horizontally adjacent basic tiles by deleting b_1 and b_2 of the right tile. To place two vertically adjacent basic tiles, we delete b_3 and b_4 of the top tile. Clearly, we have the following result.

LEMMA 2.4. *For two horizontally or vertically adjacent basic tiles, their minimum forests must have the same parity.*

Now we explain the NOT tile, which is used to change the parity. A horizontal NOT tile is shown in Figure 2.9. It is easy to check that a horizontal NOT tile can be placed between two basic tiles, according to the dimension specified in the figure. A vertical NOT tile is symmetric and can be obtained by reflecting a horizontal NOT tile with respect to line $y = x$. The parity of a minimum forest of a horizontal (vertical, respectively) NOT tile is 1 if the rightmost white point is connected by a horizontal (vertical, respectively) edge, and 0 otherwise. In Figure 2.9, since the rightmost white point q is connected by a solid vertical edge, the minimum forest that uses all solid edges has parity 0.

LEMMA 2.5. *If a horizontal NOT tile is placed between two basic tiles in a row (see Figure* 2.10*), then in the minimum forest, the parities of the two basic tiles must*

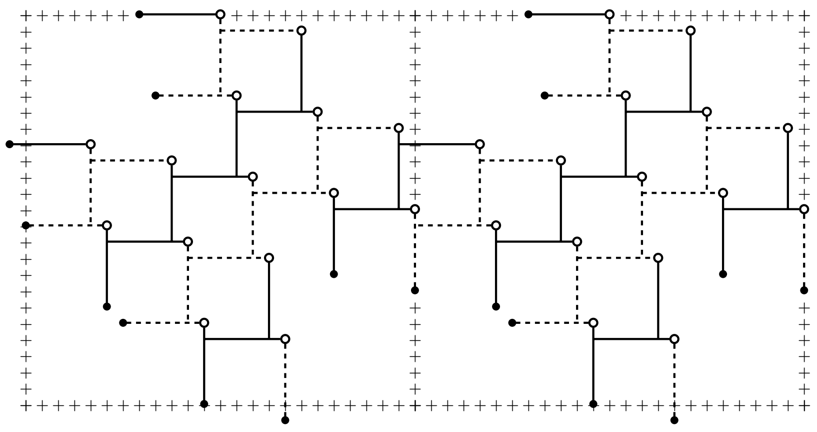

FIG. 2.8. *Two adjacent basic tiles have the same parity.*

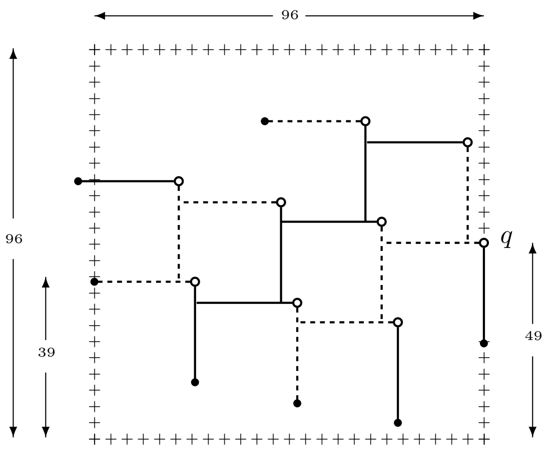

FIG. 2.9. *A horizontal NOT tile, where $\alpha = \beta = 25$, $\delta_1 = 5$, and $\delta_2 = 4$.*

be different. The same is true for the vertical case.

Proof. By observation, see Figures 2.9 and 2.10. It is easy to check that the interfaces between the basic tiles and the NOT tile satisfy the requirements of the quadruped. ☐

A clause tile is shown in Figure 2.11. It has the same interface to the left and below as a basic tile.

LEMMA 2.6. *The length of the minimum forest of a clause tile is* 108*, which is achievable only if the tile to the left has parity* 1*, or the tile below has parity* 0.

Proof. Note that q_1, q_2, q_3, and q_4 each requires an edge of length $\alpha = 20$. There are two ways to connect q_5 and q_6: through edge (q_2, q_5) and edge (q_3, q_6) of length $20 + 20 = 40$ (shown as dashed edges in Figure 2.11) or through a Steiner point of length $24 + 4 = 28$ (shown as solid edges in Figure 2.11).

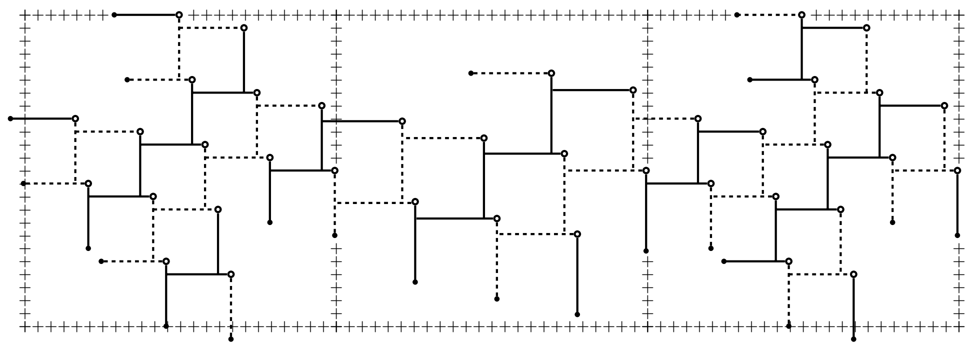

FIG. 2.10. *The parity of the left basic tile is different from the parity of the right basic tile due to the NOT tile in the middle.*

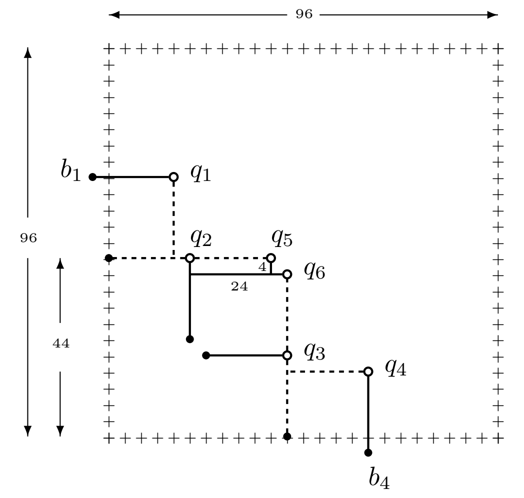

FIG. 2.11. *A clause tile, where $\alpha = \beta = 20$ and $\delta_1 = \delta_2 = 4$, unless marked otherwise.*

Therefore, the minimum forest has length 108, which is achievable only if q_2 or q_3 is connected by a solid edge. This in turn requires q_1 or q_4 to be connected by a solid edge. In order for q_1 to be connected by a solid edge, the parity of the forest on the left must be 1. In order for q_4 to be connected by a solid edge, the parity of the forest below must be 0. □

Finally, we show the OR tile in Figure 2.12. There are two parts in the OR tile: The left part acts like a clause tile, and the right part adjusts the total width to 192. The parity of a minimum forest of a OR tile is 1 if the rightmost white point q is connected by a horizontal edge, and 0 otherwise.

Note that an OR tile can always have a minimum forest of parity 0. In Figure 2.12, all white points starting with q_5 and q_6 to the right can be connected by dashed edges, regardless of what happens to the left and below. However, for the OR tile to have a minimum forest of parity 1, either q_1 or q_4 must be connected by the solid edge. Since the right side of an OR tile is always a clause tile, the OR tile will try to have parity 1 if possible in order to reduce the cost of the clause tile on the right.

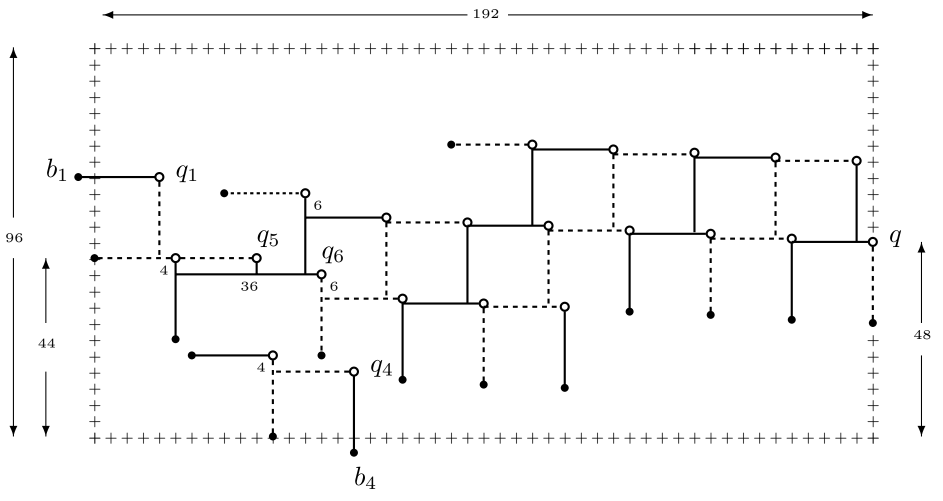

FIG. 2.12. *An OR tile, where $\alpha = \beta = 20$, $\delta_1 = 1$, and $\delta_2 = 4$, unless marked otherwise.*

LEMMA 2.7. *If the parity of a minimum forest of an OR tile is* 1*, then either the minimum forest to the left has parity* 1 *or the minimum forest below has parity* 0*. On the other hand, if either the minimum forest to the left has parity* 1 *or the minimum forest below has parity* 0*, then the parity of the OR tile can be* 1*.*

Proof. The proof is similar to the proof for the clause tile. In order for the OR tile to have parity 1, q_5 and q_6 cannot be connected by the dashed edges, which requires q_1 or q_4 to be connected by solid edges. Therefore, the minimum forest on the left must have parity 1 or the minimum forest below must have parity 0. ☐

2.4. Main theorem. We construct an RSA instance from the 3SAT instance as follows. Each variable vertex and auxiliary vertex is replaced by a basic tile, each NOT vertex is replaced by a NOT tile, each pair of OR vertices is replaced by an OR tile, and each clause vertex is replaced by a clause tile.

To connect the black points without affecting any minimum forests for the white points, we need to add additional points. From Definition 2.1 and Lemma 2.2, if there is no point in the forbidden regions, then the properties of all tiles discussed above will not be affected. Therefore we add additional black points as follows: For each black point b_i in a basic, NOT, OR, or clause tile, we add a trial of additional black points distance 1 apart to connect b_i to the nearest black/white points to the left or below, provided that the trial does not enter any forbidden region. Such a trial always exists since any forbidden region must have a white point at the upper right corner. Figure 2.13 shows how to add trials of black points to connect some black points in a basic tile.

Since the newly added black points are not in any forbidden region, these black points will not affect the minimum forests for the quadrupeds. Furthermore, since the black points have exactly one closest neighbor to the left and below, of distance 1, all black points will be connected with a fixed edge length independent of how the white points are connected. Figure 2.14 shows the RSA instance from Figure 2.5.

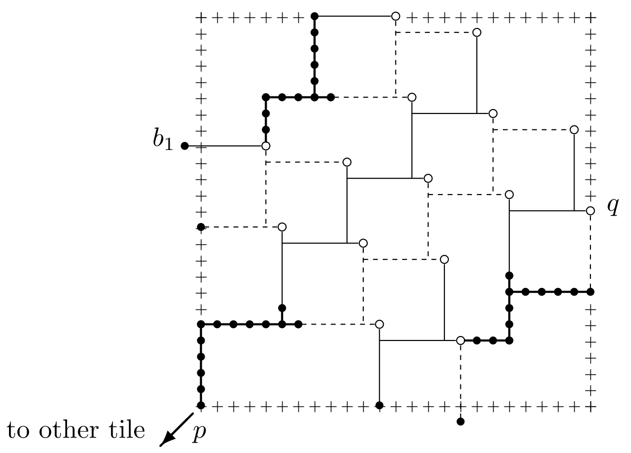

FIG. 2.13. *Trials of additional black points are added. The additional points are distance* 1 *apart and are not in any forbidden region. Therefore the black points will not affect how white points are connected.*

Some readers may wonder if graph G, H, or R contains a cycle, then whether the corresponding construction and connection will also contain a cycle. From Figure 2.13, it is easy to see that a cycle in G, H, or R does not correspond to a cycle in the connection. In the figure, the effect of the connection for q is propagated to b_1, but q is not directly connected to b_1 in the Steiner tree in this tile.

THEOREM 2.8. *The RSA problem is NP-complete in the strong sense.*

Proof. It was shown in [14] that the RSA problem has the Hanan property; hence it is in NP.

For the transformed RSA instance, let L be the sum of minimum edge lengths for connecting all black points, and for the minimum forest of each basic tile, OR tile, and NOT tile. Then we claim that the set of points has an RSA of length $L + 108m$ if and only if the planar 3SAT instance has a satisfying assignment, where m is the number of clauses.

If the 3SAT is satisfiable, then for each variable v_i we make the minimum forest for the basic tiles corresponding to v_i have parity 1 if $v_i = 1$, or parity 0 if $v_i = 0$. From Lemmas 2.3 and 2.4, the parities of the basic tiles will force the parity of other tiles. Since all clauses are satisfied, from Lemmas 2.5 and 2.6, we can use the correct minimum forest for each OR tile and clause tile. Since every clause tile has edge length 108, we will have an RSA of total edge length $L + 108m$.

On the other hand, assume there is an RSA of total edge length $L + 108m$. From Lemmas 2.2, 2.3, 2.4, and 2.5, each tile must use either the dashed edges or the solid edges, the parities of adjacent tiles must match, and all clause tiles must have length 108. From Lemma 2.6, at least one variable in each clause is true. Therefore the RSA corresponds to a true assignment.

The RSA problem is strongly NP-complete because the planar 3SAT problem is strongly NP-complete, and the coordinates used in our construction are of polynomial size. □

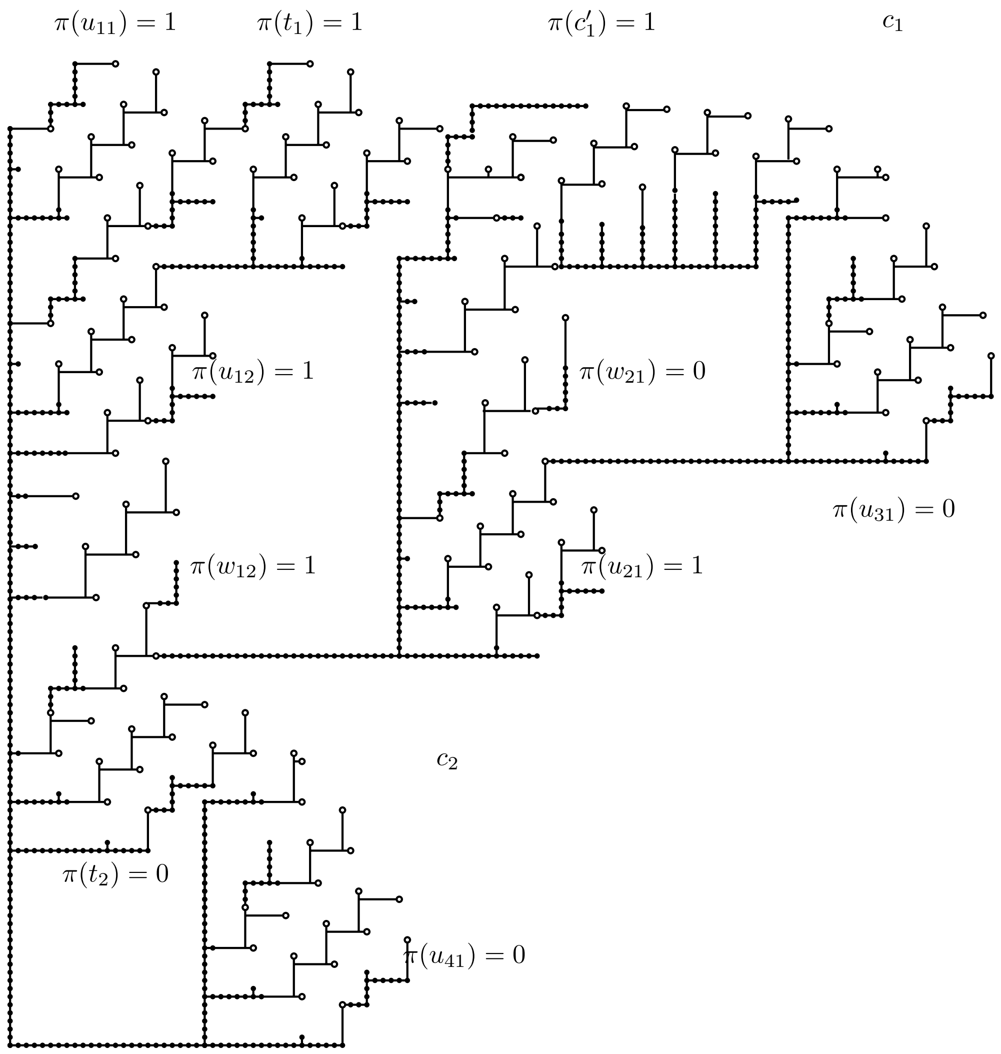

FIG. 2.14. *The transformed RSA instance, its solution, and parities of the tiles.*

Acknowledgments. The authors wish to thank Sandor Fekete and an anonymous referee for improving the presentation.

REFERENCES

[1] M. J. ALEXANDER AND G. ROBINS, *New performance-driven FPGA routing algorithms*, IEEE Trans. Computer-Aided Design, 15 (1996), pp. 1505–1517.

[2] S. ARORA, *Polynomial time approximation scheme for Euclidean traveling salesman and other geometric problems*, J. ACM, 45 (1998), pp. 753–782.

[3] J. D. CHO, *Min-cost flow based min-cost rectilinear Steiner distance preserving tree construction*, in Proceedings of the International Symposium on Physical Design, ACM Press, New York, 1997, pp. 82–87.

[4] J. Cong, A. B. Kahng, and K. S. Leung, *Efficient algorithms for the minimum shortest path Steiner arborescence problem with applications to VLSI physical design*, IEEE Trans. Computer-Aided Design, 17 (1998), pp. 24–39.

[5] J. Cong, K. S. Leung, and D. Zhou, *Performance driven interconnect design based on distributed RC delay model*, in Proceedings of the 30th ACM/IEEE Design Automation Conference, 1993, pp. 606–611.

[6] J. Córdova and Y. H. Lee, *A Heuristic Algorithm for the Rectilinear Steiner Arborescence Problem*, Technical Report TR-94-025, Department of Computer Science, University of Florida, Gainesville, FL, 1994.

[7] A. Erzin and A. Kahng, *private communication.*

[8] M. R. Garey and D. S. Johnson, *Computers and Intractability. A Guide to the Theory of NP-Completeness*, W. H. Freeman, San Francisco, 1979.

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

[10] R. R. Laderia de Matos, *A Rectilinear Arborescence Problem*, Ph.D. dissertation, University of Alabama, Tuscaloosa, AL, 1979.

[11] D. Lichtenstein, *Planar formulae and their uses*, SIAM J. Comput., 11 (1982), pp. 329–343.

[12] B. Lu and L. Ruan, *Polynomial time approximation scheme for rectilinear Steiner arborescence problem*, J. Comb. Optim., 4 (2000), pp. 357–363.

[13] L. Nastansky, S. M. Selkow, and N. F. Stewart, *Cost-minima trees in directed acyclic graphs*, Z. Operations Res. Ser. A-B, 18 (1974), pp. 59–67.

[14] S. K. Rao, P. Sadayappan, F. K. Hwang, and P. W. Shor, *The rectilinear Steiner arborescence problem*, Algorithmica, 7 (1992), pp. 277–288.

[15] V. A. Trubin, *Subclass of the Steiner problems on a plane with rectilinear metric*, Cybernetics, 21 (1985), pp. 320–322.

[16] L. Valiant, *Universality considerations in VLSI circuits*, IEEE Trans. Comput., 30 (1981), pp. 135–140.