COMPUTING HEXAGONAL STEINER TREES USING PCX

Andrew P. Thurber and Guoliang Xue

Department of Computer Science
The University of Vermont
Burlington, VT 05405-0156
Email: {athurber,xue}@cs.uvm.edu

1. INTRODUCTION

An important problem in VLSI physical design is to compute the shortest network interconnecting a given set of terminal points on the plane. In traditional VLSI design, wires are allowed to go horizontally and vertically, which leads to the rectilinear Steiner minimum tree problem. Recent technology advancement has made it possible for wires to go in three or four directions. These lead to the hexagonal Steiner minimum tree problems (where wires are allowed to go in directions making angles of 0^o or 60^o with the x-axis) and Steiner minimum tree problems with 45^o routing (where wires are allowed to go in directions making angles of 0^o, 45^o or 90^o with the x-axis). All of the above Steiner minimum tree problems are NP-hard [7, 5]. However, due to their important CAD applications, these problems have received more and more attention in the literature [4, 7, 8, 9, 10, 11, 12, 13, 16, 14, 21]. Several heuristic algorithms have been proposed.

Due to the lack of an exact algorithm for computing optimal solutions, researchers have to compare the hexagonal cost of heuristic trees with the cost of the Euclidean Steiner minimum trees [9, 10]. This paper provides an exact algorithm which can compute optimal solutions for small-sized problems. Such an exact algorithm can be used to measure the performance of heuristic algorithms and can also be used in the design of good heuristic algorithms.

The rest of this paper is organized as follows. In Section 2, we show that the hexagonal distance between two points is the maximum of six affine functions of the coordinates of the two points. In Section 3, we will show that the shortest hexagonal network under a given full Steiner topology and its degeneracies can be solved as a linear programming problem. In Section 4, we present a simple method to enumerate all of the full Steiner topologies and prove a bounding theorem which can be used for effective pruning in a branch and bound algorithm. In Section 5, we present some computational results. We conclude this paper in Section 6.

This research was supported in part by the Army Research Office grant DAAH04-96-1-0233 and by the NSF grants ASC-9409285 and OSR-9350540. All correspondence should be addressed to G.L. Xue.

2. THE HEXAGONAL DISTANCE $\lambda_3(\bullet,\bullet)$

The λ_3 distance function was introduced by Widmayer, Wu and Wong [16] and has been extensively studied in [4, 7, 8, 21]. A line segment is *legal* if it makes an angle of 0^o or 60^o with the x-axis. An edge $e(p_i,p_j)$ connecting two points p_i and p_j is a shortest zig-zaging line connecting p_i and p_j with every segment being legal. It has been proved that an edge can be made either straight or consisting of two segments making an angle of 120^o [7, 16]. The hexagonal distance between p_i and p_j (denoted by $\lambda_3(p_i,p_j)$) is the length of the edge $e(p_i,p_j)$.

Lemma 2.1 *Let $p_i=(x_i,y_i)$ and $p_j=(x_j,y_j)$ be two points in the plane. Then the hexagonal distance between p_i and p_j can be computed using the following formula.*

$$\lambda_3(p_i,p_j)=\max\{\frac{2}{\sqrt{3}}|y_j-y_i|,|x_j-x_i|+\frac{1}{\sqrt{3}}|y_j-y_i|.$$

PROOF. Follows directly from the works in [7, 16]. □

Note that once the pairwise *connections* among the terminal points and the Steiner points are given, the *optimal locations* for the Steiner points under the given connections can be computed by solving a linear programming problem, since the cost of the tree is the sum of edge lengths. This will be formalized in the next section.

3. SHORTEST NETWORK UNDER A STEINER TOPOLOGY

Definition 3.1 *Let $P=\{p_1,p_2,\ldots,p_n\}$ be a set of n terminal points in the plane. A* Steiner topology *interconnecting P is a tree network $T=(V,E)$ such that*

1. *The set of vertices V can be decomposed into two disjoint sets V_P and V_S where $V_P=\{v_1,\ldots,v_n\}$ are the* terminal vertices *such that v_j corresponds to the terminal point p_j whose location is given $(j=1,2,\ldots,n)$; and that $V_S=\{v_{n+1},v_{n+2},\ldots,v_{n+m}\}$ are the* Steiner vertices *such that v_{n+j} corresponds to a Steiner point s_{n+j} whose location can be moved $(j=1,2,\ldots,m)$.*
2. *Every degree-1 vertex is a terminal vertex. Every Steiner vertex has degree at least 3.*

0-7803-5682-9/99/$10.00©1999 IEEE.

A Steiner topology $\mathcal{T}$ is called a full Steiner topology *if the degree of each terminal vertex is exactly* 1 *and the degree of each Steiner vertex is exactly* 3.

Definition 3.2 *Let $\mathcal{T}$ be a Steiner topology. An edge is called a* Steiner-terminal edge *if exactly one of its vertices is a terminal vertex (the other an Steiner vertex). Let $e = \{u, v\}$ be a Steiner-terminal edge where u is a terminal vertex corresponding to a terminal point p, we can* shrink *this edge by deleting the edge e and replacing the two vertices u and v by a new terminal vertex which corresponds to the terminal point p. Any vertex other than u and v is connected to this new vertex if and only if it was connected to u or v before the shrinking operation. Notice that a Steiner topology changes to another Steiner topology when a Steiner-terminal edge is shrunk, provided that the degree of the resulting new vertex is no more than* 3. *A Steiner topology T' is called a* degeneracy *of T if T' can be obtained from T using zero or more number of shrinking operations. We use $D(T)$ to denote the set of Steiner topologies which are degeneracies of T.*

Definition 3.3 *A* network *T under a given Steiner topology $\mathcal{T}$ is obtained by fixing every terminal vertex v_j in $\mathcal{T}$ at the location of its corresponding terminal point $l(v_j) = p_j$ $(j = 1, \ldots, n)$ and fixing every Steiner vertex v_{n+j} in $\mathcal{T}$ at some location $l(v_{n+j})$ in the plane $(j = 1, \ldots, m)$. The* cost *of a network T is defined as the sum of edges costs where the cost of each edge is measured as the hexagonal distance between the locations of its two end vertices:*

$$cost(T) = \sum_{(v_i, v_j) \in E} \lambda_3(l(v_i), l(v_j)). \qquad (1)$$

A network under $\mathcal{T}$ with the minimum cost is called a shortest network under topology $\mathcal{T}$. *A* shortest network under $D(\mathcal{T})$ *is a minimum cost network under a topology in $D(\mathcal{T})$.*

Theorem 3.1 *Let $\mathcal{T}$ be a full Steiner topology interconnecting P. Then the shortest network under $D(\mathcal{T})$ can be computed by solving the following linear programming problem.*

$$\begin{aligned} \min \quad & \sum_{(v_i, v_j) \in E} \beta_{i,j} \qquad (2) \\ \text{s. t.} \quad & \beta_{i,j} \geq \tfrac{2}{\sqrt{3}}(y_j - y_i), \\ & \beta_{i,j} \geq \tfrac{2}{\sqrt{3}}(y_i - y_j), \\ & \beta_{i,j} \geq (x_j - x_i) + \tfrac{1}{\sqrt{3}}(y_j - y_i), \\ & \beta_{i,j} \geq (x_j - x_i) + \tfrac{1}{\sqrt{3}}(y_i - y_j), \\ & \beta_{i,j} \geq (x_i - x_j) + \tfrac{1}{\sqrt{3}}(y_j - y_i), \\ & \beta_{i,j} \geq (x_i - x_j) + \tfrac{1}{\sqrt{3}}(y_i - y_j), \end{aligned}$$

where (x_i, x_j) is the location of the terminal point p_j which corresponds to terminal vertex $v_j (j = 1, 2, \ldots, n)$ and (x_{n+j}, y_{n+j}) is the location of Steiner vertex v_{n+j} $(j = 1, 2, \ldots, n-2)$.

PROOF. Note that in our linear programming formulation, $\beta_{i,j}$ is an upper bound on the λ_3 length of edge $(v_i, v_j) \in E$. Since the objective is to minimize the sum of $\beta_{i,j}$ over all edges $(v_i, v_j) \in E$, $\beta_{i,j}$ equals the λ_3 length of edge (v_i, v_j). This proves the theorem. □

4. THE BRANCH AND BOUND ALGORITHM

In [15], Smith proved the following result which establishes a one-to-one correspondence between the set of full Steiner topologies on n terminal points and a special set of $(n-3)$-element vectors.

Lemma 4.1 *There is a one-one correspondence between full Steiner topologies on $n \geq 4$ terminal points, and $(n-3)$-element vectors $t = (t_4, t_5, \ldots, t_n)$ whose ith entry t_i is an integer in the range $1 \leq t_i \leq 2i-5$. Therefore the number of full Steiner topologies on n terminal points is $3 \cdot 5 \cdots (2n-5)$.* □

An illustration of this one-to-one correspondence can be found in [19]. A brute-force algorithm for computing a hexagonal Steiner minimum tree is to compute the minimum cost network under a full Steiner topology for every full Steiner topology interconnecting the n terminal points. Since there are $3 \times 5 \times \cdots \times (2n-5)$ different full Steiner topologies for a set of n fixed points, a complete enumeration method is very expensive. As in [19], we can prove a similar bounding theorem to ensure that the following branch-and-bound algorithm works correctly.

Algorithm 1 [Branch and Bound]

Step_1 Compute an upper-bound UB using a good heuristic algorithm. Set $k := 4$ and $t_4 := 3$.

Step_2 Compute the minimum cost network under the topology with topological vector $(t_4, t_5, \ldots, t_k)$. Let C be the cost of the current network.

Step_3 **if** $C \geq UB$ **then goto** Step_4 **else goto** Step_5 **endif**

Step_4 **if** $t_k > 1$ **then**
  $t_k = t_k - 1$; **goto** Step_2;
  else if $k = 4$ **then**
  stop ;
  UB is the minimum cost and $(b_4, b_5, \ldots, b_n)$ is the optimal topological vector.
  else
  $k := k - 1$; **goto** Step_4;
  endif

Step_5 **if** $k = n$ **then**
  Set $UB := C$ and save the current topological vector in $(b_4, b_5, \ldots, b_n)$;
  goto Step_4;
  else
  $k := k + 1$; $t_k = 2k - 5$; **goto** Step_2;
  endif

5. ENHANCING THE PERFORMANCE

Our branch-and-bound algorithm presented in the previous section can find a hexagonal Steiner minimum tree for up to 15 terminal points within 20 minutes on a Pentium based PC. As with all branch and bound algorithms, it becomes too expensive for larger problems. In this section, we address three important issues for solving large-scale hexagonal Steiner minimum tree problems. These are the min-min heuristic for initial upper-bound and the max-min heuristic for effective pruning.

The min-min heuristic is a natural generalization of Prim's algorithm for computing a minimum spanning tree [2]. Here, instead of growing a spanning tree, we are going a full Steiner topology. At the very beginning, our topology consists of a single edge which connects a pair of terminal points with the shortest hexagonal distance. Other terminal points are added to the topology one by one using the following greedy heuristic:

Let T be the current full Steiner topology interconnecting a subset of k of the n terminal points. Let p be a terminal point not already in the topology. Connecting p to any of the $2k-3$ edges would result in a full Steiner topology with terminal p joining the previous topology. Since T has $2k-3$ edges, there are $2k-3$ way for p to join the current topology. The *joining cost* of p is the minimum cost of the $2k-3$ possible topologies. The terminal with the minimum joining cost will join the topology at each step of the min-min heuristic algorithm. It is clear that the min-min heuristic requires $O(n^3)$ PCx calls.

In the branch and bound algorithm, a branch is cutoff only if its associated cost is no less than the cost of the current incumbent. We have used a max-min heuristic ordering of the terminal points so that a cutoff is likely to happen high in the search tree. The max-min heuristic also grows a full Steiner topology by adding one terminal point at each step. In this heuristic, the initial topology consists of an edge connecting a pair of terminals with the longest hexagonal distance. Using the same *joining cost* as defined in the min-min heuristic, the terminal with the *maximum* joining cost joins the topology at each step of the max-min heuristic. It is clear that max-min and min-min have the same time complexity. The max-min algorithm also produces an ordering of the terminal points (in the order they join the topology). The idea behind the max-min heuristic is to lay out a good frame for the terminals that are far away from each other. This idea has been used in [17, 18]. Computational experiences in [17, 18] show that this kind of heuristics improve the performance of the branch-and-bound algorithm dramatically.

6. COMPUTATIONAL RESULTS

We have implemented our branch and bound algorithm (along with several heuristics) in ANSI C and tested it on a Pentium 450 MHz running RedHat Linux. As in [9, 10], test problems were taken from the OR-library [1]. We have computed the hexagonal Steiner minimum trees for all 15 instances of estein10 and for 14 of the 15 instances of of estein20.

Index	L(MST)	L(SMT)	PCx-C	CPU(sec)
1	2.3679	2.2039	868	48.71
2	1.8176	1.7905	127	8.37
3	2.6193	2.3906	110	5.56
4	2.0253	1.9978	230	10.58
5	1.8911	1.8509	80	2.11
6	2.6158	2.4161	180	8.67
7	2.5599	2.4306	477	14.19
8	2.4019	2.3169	146	5.93
9	2.1806	2.0770	673	29.20
10	2.2646	2.2623	518	31.62
11	2.2764	2.1274	508	20.86
12	1.9657	1.9616	251	10.49
13	2.0313	1.8672	101	6.04
14	2.3163	2.1141	339	10.49
15	1.9348	1.8395	700	36.33
average			353.9	16.61

Table 1: Branch and bound results for estein10.

Index	L(MST)	L(SMT)	PCx-C	CPU(hrs)
1*	3.528465	3.365920	206.19e6	3693.9
2	3.154736	3.054010	0.06e6	0.8
3	2.740162	2.702046	1.66e6	26.9
4	2.815312	2.700635	1.20e6	19.1
5	3.413392	3.190313	4.00e6	61.7
6	3.601201	3.426707	0.11e6	2.0
7	3.413916	3.288207	1.48e6	23.3
8	3.787465	3.634126	16.86e6	275.7
9	3.523634	3.511853	2.68e6	44.4
10	3.363949	3.226969	5.98e6	101.3
11	2.650403	2.531834	0.25e6	4.0
12	3.017058	2.874737	25.06e6	113.1
13	3.409348	3.317156	1.38e6	22.6
14	3.300822	3.200889	3.85e6	60.5
15	3.103847	2.965345	0.43e6	7.4
average(instances 2-15)			4.6e6	54.49

Table 2: Branch and bound results for estein20.

Table 1 shows the results for estein10. The first column shows the instance index in estein10. The second column is the cost of a minimum spanning tree. The third column is the cost of the hexagonal Steiner minimum tree. The last column shows the CPU seconds used. The column with index pcx-C is the number of PCx calls. Note that the number of full Steiner topologies involving smaller than 10 terminals is 146599 and

the number of full Steiner topologies involving 10 terminals is 2027025. It is clear that our branch and bound algorithm is only searching a very small portion of the search space. Table 2 shows the results for estein20. We have obtained the optimal solutions for instances 2–15. However, our algorithm failed to find the optimal solution for instance 1 within the time available. The cost presented here is only an upper bound of the cost of the optimal solution.

7. CONCLUSIONS

In this paper, we have presented a branch and bound algorithm for computing a hexagonal Steiner minimum tree, where the fixed-topology problems are solved using PCx. Although currently it can only compute optimal solutions for small-sized instances, this algorithm makes it possible for researchers to compare heuristic solutions against *optimal* solutions for small-sized instances. Our min-min and max-min heuristics have also been shown to be good heuristic algorithms for computing hexagonal Steiner minimum trees, but not as good as the iterated 1-Steiner heuristic [11]. In [13], we have designed a linear time algorithm for computing an optimal layout of a minimum spanning tree, which can be viewed as a fast approximation to hexagonal Steiner minimum trees.

8. REFERENCES

[1] J.E. Beasley, OR-library: distributing test problems by electronic mail, *Journal of Operational Research Society*, 41(1990): 1069–1072.

[2] T.H. Cormen, C.E. Leiserson and R.L. Rivest, *Introduction to Algorithms*, MIT Press, 1990.

[3] J. Czyzyk, S. Mehrotra, M. Wagner and S.J. Wright, *PCx User Guide (Version 1.1)*, TechReport OTC96/01, Optimization Technology Center, Argonne National Lab and Northwestern University.

[4] D.Z. Du and F.K. Hwang, Reducing the Steiner problem in a normed space, *SIAM Journal on Computing*, 21(1992): 1001–1007.

[5] F.K. Hwang, D.S. Richards, and P. Winter, *The Steiner tree problems*, Annals of Discrete Mathematics, Vol. 53(1992), North-Holland.

[6] A.B. Kahng and G. Robins, A new class of iterative Steiner tree heuristics with good performance, *IEEE Trans. on CAD*, 11(1992): 893–902.

[7] D.T. Lee and C.-F. Shen, The Steiner minimal tree problem in the λ-geometry plane, *ISAAC'96*: 247–255.

[8] D.T. Lee, C.-F. Shen and C.-L. Ding, On Steiner tree problem with 45^o routing, *ISCAS'95*: 1680–1683.

[9] Y.Y. Li, S.K. Cheung, K.S. Leung and C.K. Wong, On the Steiner tree problem in λ_3-metric, *ISCAS'97*: 1564–1567.

[10] Y.Y. Li, S.K. Cheung, K.S. Leung and C.K. Wong, Steiner tree constructions in λ_3-metric, *IEEE Trans. Circuits and Systems II*, 45(1998), 563–574.

[11] G.-H. Lin, A. Thurber and G.L. Xue, The 1-Steiner tree problem in λ_3 geometry plane, accepted by *ISCAS'99*.

[12] G.-H. Lin and G.L. Xue, The Steiner tree problem in λ_4 plane, *ISAAC'98*: 327–336.

[13] G.-H. Lin and G.L. Xue, A linear time algorithm for optimal layout of a minimum spanning tree in λ_3 geometry plane, *Manuscript*, 1999.

[14] M. Sarrafzadeh and C.K. Wong, Hierarchical Steiner tree construction in uniform orientations, *IEEE Trans. on CAD*, 11(1992): 1095–1103.

[15] W.D. Smith, How to find Steiner minimal trees in Euclidean d-space, *Algorithmica*, 7(1992): 137–177.

[16] P. Widmayer, Y.F. Wu and C.K. Wong, On some distance problems in fixed orientations, *SIAM Journal on Computing*, 16(1987): 728–746.

[17] G.L. Xue, G.-H. Lin and D.Z. Du, Grade of service Euclidean Steiner minimum trees, accepted by *ISCAS'99*.

[18] G.L. Xue, T.P. Lillys and D.E. Dougherty, Computing the minimum cost pipe network interconnecting one sink and many sources, *SIAM Journal on Optimization*, accepted for publication.

[19] G.L. Xue, Y.Y. Ye, An efficient algorithm for minimizing a sum of Euclidean norms with applications, *SIAM Journal on Optimization*, 7(1997), pp. 1017-1036.

[20] G.L. Xue, Y.Y. Ye, An efficient algorithm for minimizing a sum of p-norms, *SIAM Journal on Optimization*, accepted for publication.

[21] G.Y. Yan, A. Albrecht, G.H.F. Young and C.K. Wong, The Steiner tree problem in orientation metrics, *Journal of Computer and System Sciences*, 55(1997), 529–546.