Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 EECS Building
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 00-059

A Polynomial Time Approximation Scheme for the problem of Interconnecting Highways

Xiuzhen Cheng, Joon-mo Kim, and Bing Lu

November 28, 2000

A Polynomial Time Approximation Scheme for the problem of Interconnecting Highways

Xiuzhen Cheng Joon-Mo Kim Bing Lu
Department of Computer Science and Engineering
University of Minnesota
Minneapolis, MN 55455, U.S.A.
E-mail: {cheng, jkim, blu}@cs.umn.edu

November 15, 2000

Abstract

The objective of the Interconnecting Highways problem is to construct roads of minimum total length to interconnect n given highways under the constraint that the roads can intersect each highway only at one point in a designated interval which is a line segment. We present a polynomial time approximation scheme for this problem by applying Arora's framework in [2]. For every fixed $c > 1$ and given any n line segments in the plane, a randomized version of the scheme finds a $(1 + \frac{1}{c})$-approximation to the optimal cost in $O(n^{O(c)}log(n))$ time.

Key words and phrases. Interconnecting highways, PTAS, one-way truncating, two-way truncating, k-Patching.

1 Introduction

The Interconnecting Highways problem asks for a shortest network consisting of *roads* with minimum total length to interconnect n highways $H_1, H_2, \cdots, H_n$ under the constraint that the roads can intersect H_i only at one point, called an *exit* of H_i, in a designated interval I_i. For simplicity, we assume that all I_i's are disjoint. We are interested in the case when all intervals are line segments, including the extreme case that each I_i is a point. This is the generalized Minimum Steiner Tree problem (SMT). When all intervals are points, the problem is reduced to the traditional SMT problem which is an active combinatorial optimization problem [10, 5, 1, 7]. Thus it is NP-Hard, since SMT is NP-Hard [8].

Chen [4] and Weng [13] studied the special case when $n = 2$. Du *et al.* [6] first tackled the optimality conditions of this problem and then gave a method to construct a solution to meet this set of conditions by generalizing Melzak's construction for Steiner trees. Du *et al.* [6] also determined the optimal solutions for $n = 2$ and $n = 3$. Given a full Steiner topology, Xue *et al.* [14] computed an ϵ-optimal solution to this problem

in $O\left(n\sqrt{n}log(n) + n\sqrt{n}log\left(\frac{\bar{s}}{\epsilon}\right)\right)$ arithmetic operations, where $\bar{s}$ is the largest pair-wise distance of the midpoints of the given line segments.

An algorithm A is a Polynomial Time Approximation Scheme (PTAS) for a minimization problem with optimal cost OPT if the following is true: Given an instance I of the problem and a small positive error parameter ϵ, (i) the algorithm outputs a solution which is at most $(1+\epsilon)\,OPT$; (ii) when ϵ is fixed, the running time is bounded by a polynomial in the size of the instance I. The design of all PTAS's is based on the idea of trading accuracy for running time. Based on the error parameter ϵ, the given problem instance is transformed to a coarser instance which can be solved by dynamic programming. If there exists a PTAS for an optimization problem, the problem's instance can be approximated to any required degree.

For convenience, we rephrase the problem of Interconnecting Highways in the following way. Given n disjoint line segments in the plane, find a Steiner tree consisting of roads as edges to connect all segments such that the sum of road length (the total length of all roads in the tree, denoted by L_T) and segment length (the total length of all given segments, denoted by L_S) is minimized under the constraint that each segment has exactly one point (the exit) to be connected to the tree. If all segments are points, this becomes the Minimum Steiner Tree (MST) problem in Euclidean plane. Note that in the rephrased problem we choose $L_T + L_S$ instead of L_T as the objective function. In the optimal case, there is no difference since L_S is a constant. When we work on some approximate algorithm, L_S must be bounded to get a reasonable approximation for L_T. Here is why. Let $OPT_{L_T} + L_S$ be the optimal cost and $APP_{L_T} + L_S$ be an approximation with performance ratio ϵ, where ϵ is a positive real number. Then, $APP_{L_T} + L_S \leq (1+\epsilon)\,(OPT_{L_T} + L_S)$. Thus we have, $APP_{L_T} \leq (1+\epsilon)\,OPT_{L_T} + \epsilon\,L_S$. If L_S is upper bounded by $c' \cdot LB$ for some small constant real number c', where LB is the lower bound of OPT_{L_T}, then $APP_{L_T} \leq (1+\epsilon')OPT_{L_T}$, where $\epsilon' = \epsilon \cdot (1+c')$.

From now on, unless otherwise specified, we use Steiner tree to refer to a tree whose edge set consists of both line segments and roads. A minimum Steiner tree is the Steiner tree with minimum $L_T + L_S$. Note that we have three kinds of nodes and two kinds of edges in the Steiner tree. Node set includes *endpoints of line segments*, *exits in line segments*, and *real Steiner points with degree 3* [6]. Edge set includes *road edges* and *segment edges*. A Steiner tree is *valid* if it connects to each line segment at exactly one point.

This paper provides a PTAS for the Interconnecting Highways problem. We find a $(1+\frac{1}{c})$-approximation to the optimum in time $O(n^{O(c)}log\,n)$ by applying Arora's framework in [1, 2] for some constant $c > 1$. The given problem instance is first perturbed and rescaled, making it well-rounded. Then, dynamic programming technique is applied to compute a $(1+\frac{1}{c})$-approximate result. The correctness of this algorithm is evidenced by Structure Theorem, which constructively proves the existence of the polynomial time approximation scheme.

2 A PTAS for the Problem of Interconnecting Highways

As mentioned earlier, we will apply Arora's technique [1, 2] to the rephrased Interconnecting Highways problem. The basic idea of the PTAS is sketched below. Given an instance of n line segments and a constant $c > 0$, the Structure Theorem (Theorem 2.2) guarantees that the square containing all line segments can be recursively partitioned such that some $\left(1 + \frac{1}{c}\right)$-approximate Steiner tree crosses each line of the partition at most $2r$ times, where $r = O(c)$, and all these crosses happen at some prespecified points (*portals* for road edges and cross points for segment edges) in the line. Such a Steiner tree can be found by dynamic programming. In other words, we decompose the instance of our problem into a large number of "independent" and smaller instances and solve the original problem by dynamic programming.

2.1 Perturbation

We first perturb and rescale the instance, making it well-rounded. A well-rounded instance satisfies the following conditions: (i) the coordinates of segment endpoints and Steiner points are integral; (ii) the minimum non-zero internode distance is 4; (iii) the maximum internode distance is $O(n)$. Here nodes refer to segment endpoints or Steiner points.

Let L_0 be the size of the smallest axis-aligned bounding square and OPT be the optimal cost for the given instance, then $OPT \geq L_0$. Let $c > 1$ be a constant. We place a grid of granularity $\frac{L_0}{24\,n\,c}$ and move each segment endpoint and Steiner point to its nearest grid point. Note that each point is moved by a distance of at most $\frac{L_0}{24\,n\,c}$. The exit point in each segment moves correspondingly with the segment and thus is moved by a distance of at most $\frac{L_0}{24\,n\,c}$. Since the optimal Steiner tree contains at most $n-2$ Steiner points, there are at most $2\,n-3$ roads. Plus the n segments, the total number of edges in the optimal tree is at most $3\,n-3$. Because each edge has 2 endpoints and each point is moved by a distance of at most $\frac{L_0}{24\,n\,c}$, the cost of the tree is changed by at most $2 \cdot 3\,n \cdot \frac{L_0}{24\,n\,c} = \frac{L_0}{4\,c} \leq \frac{OPT}{4\,c}$. Rescaling distances by $\frac{L_0}{96\,n\,c}$, the instance becomes well-rounded.

Remarks: (i) Even though we have an exit which connects to the network in each segment, we treat each segment as one edge, not two edges. We do not move exits to their nearest grid points, thus we can keep the integrality of each line segment. Exits move with their associated line segments. (ii) The purpose of perturbation will be explained latter in the construction of shifted dissection and quadtree. The rescalling of internode distances helps to make Lemma 2.6 applicable in the proof of Structure Theorem. (iii) The perturbation takes $O(n\,log(n))$ time since each node needs $O(log(n))$ time to find its nearest grid point. (iv) After perturbation, all endpoints of line segments and Steiner points lie at grid points. The size of the bounding square becomes $O(24\,n\,c) = O(n)$.

Lemma 2.1 *With the perturbation described above, a PTAS for the perturbed problem is a PTAS for the original problem.*

Proof. Let OPT_{orig} be the optimal cost to the original problem, $OPT_{perturb}$ be the optimal cost to the perturbed problem. Let OPT_{grid} be the cost of the Steiner tree generated by

directly perturbing an optimal tree. A PTAS for the perturbed problem means there exists a $(1+\frac{1}{c_1})$-approximation with cost $A_{perturb}$ to the perturbed problem with polynomial running time when $c_1 > 1$ is fixed. That is, $A_{perturb} \leq \left(1+\frac{1}{c_1}\right) OPT_{perturb}$. Since $|\, OPT_{orig} - OPT_{grid}\,| \leq \frac{OPT_{orig}}{4\,c}$ and $OPT_{perturb} \leq OPT_{grid}$, we have

$$\begin{aligned} A_{perturb} &\leq \left(1+\tfrac{1}{c_1}\right) OPT_{perturb} \\ &\leq \left(1+\tfrac{1}{c_1}\right) OPT_{grid} \\ &\leq \left(1+\tfrac{1}{c_1}\right) \left(1+\tfrac{1}{4\,c}\right) OPT_{orig} \end{aligned}$$

If we move each segment endpoint in $A_{perturb}$ to its original position, we get the approximate solution A_{orig} to the original problem. Since there are at most $n-2$ Steiner points in $A_{perturb}$, similarly we have $|A_{orig} - A_{perturb}| \leq \frac{A_{perturb}}{4\,c}$. Thus

$$\begin{aligned} A_{orig} &\leq \left(1+\tfrac{1}{4\,c}\right) A_{perturb} \\ &\leq \left(1+\tfrac{1}{c_1}\right) \left(1+\tfrac{1}{4\,c}\right)^2 OPT_{orig} \\ &\leq \left(1+\tfrac{1}{c'}\right) OPT_{orig} \end{aligned}$$

where $c' > 1$ is a constant since both c and c_1 are constants.

Perturbation takes near-linear time. When c_1 is fixed, $A_{perturb}$ takes polynomial time. Thus the total running time is polynomial when c' is fixed. ■

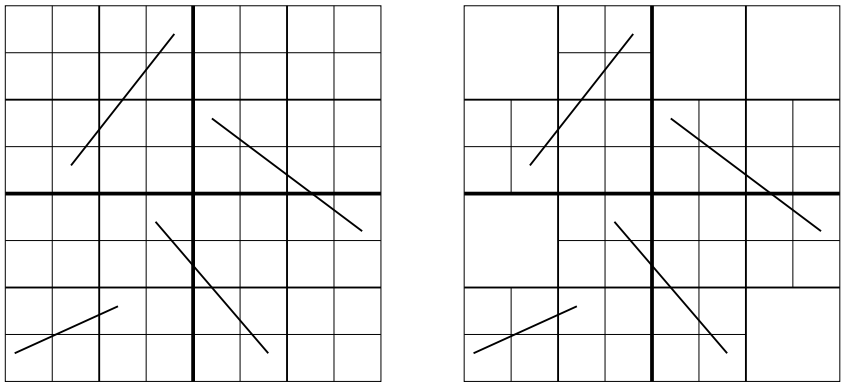
2.2 Shifted Dissection and Quadtree

As mentioned earlier, we will recursively partition the bounding square by using a randomized variant of quadtree [3] to find approximation for the perturbed instance. After perturbation, the bounding square has size $L = O(n)$. W.l.o.g. we assume L is a power of 2.

We call a recursive partion of the bounding square *a dissection*. Assume the lowerleft corner of the axis-aligned bounding square is in coordinates $(0, 0)$. Initially we place a horizontal line ($y = \frac{L}{2}$) and a vertical line ($x = \frac{L}{2}$) to divide the bounding square into 4 equal-sized smaller squares. Then we place two horizontal lines ($y = \frac{L}{4}$ and $y = \frac{3\,L}{4}$) and two vertical lines ($x = \frac{L}{4}$ and $x = \frac{3\,L}{4}$) to divide the 4 smaller squares into 16 much more smaller squares. ... We continue this procedure until the size of the newly-generated equal-sized square is ≤ 1. We view this dissection as a 4-ary *dissection tree* whose root is the bounding square. Each square is partitioned into 4 equal-sized squares which are its children. The depth of the tree equals the number of recursions of the partition which is $O(log(L)) = O(log(n))$ since $L = O(n)$. The number of leaf nodes in the dissection tree is $O(L^2) = O(n^2)$.

The construction of a *quadtree* is similar to that of a dissection tree except that we stop partitioning a square if it contains no line segment. In other words, each non-leaf node in a

quadtree contains at least one (part of) line segment while each leaf node is either empty, or if it is nonempty, its size must be ≤ 1. The depth of the quadtree is $O(log(n))$. Since the number of leaves can be $O(n^2)$, the total number of squares in a quadtree is $O(n^2\, log(n))$. Figure 1 demonstrates how a dissection and the corresponding quadtree look like.



(a) A dissection tree (b) The corresponding quadtree

Figure 1: *(a) A dissection tree. We stop the partition when the size of the square is* ≤ 1. *(b) The corresponding quadtree. Each leaf square of the quadtree is either empty, or nonempty, but with size* ≤ 1

Let a, b be integers in $[0, L)$. For the previously defined dissection, by shifting all horizontal lines from $(y = y_i)$ to $(y = y_i + b \bmod L)$ and all vertical lines from $(x = x_i)$ to $(x = x_i + a \bmod L)$, where $i = 0, 1, \cdots, L-1$, we get the *dissection with shift* (a, b). Now the horizontal line $(y = \frac{L}{2})$ is shifted to $(y = \frac{L}{2} + b \bmod L)$; The vertical line $(x = \frac{L}{2})$ is shifted to $(x = \frac{L}{2} + a \bmod L)$.

Note that dissection is just a recursive partition of the bounding square. You can understand the dissection with shift (a, b) in the following way: The dissection partitions the bounding square into many smaller squares. Each smaller square has a fixed position inside the bounding square. We push the bounding square in x direction a units. Then push the bounding square in y direction b units. The lowerleft corner of the bounding square is changed from $(0, 0)$ to (a, b). All smaller squares inside the bounding square move accordingly. Now the bounding square may not be the bounding square to the set of given line segments. Thus we "wrap around" the bounding square such that all segments are covered. Look the "wrapped-around" bounding square becomes a disjoint union of 2 (either $a = 0$ or $b = 0$) or 4 (both $a \neq 0$ and $b \neq 0$) rectangles. Correspondingly some smaller squares inside the bounding box are "wrapped-around" and becomes a disjoint union of 2 or 4 rectangles since they have fixed relative position to the bounding square. Figure 2 demonstrates how the dissection with shift (a, b) is generated. We also view a shifted dissection as a *dissection tree with shift* (a, b).

The *quadtree with shift* (a, b) is defined similarly except that when a square in the shifted

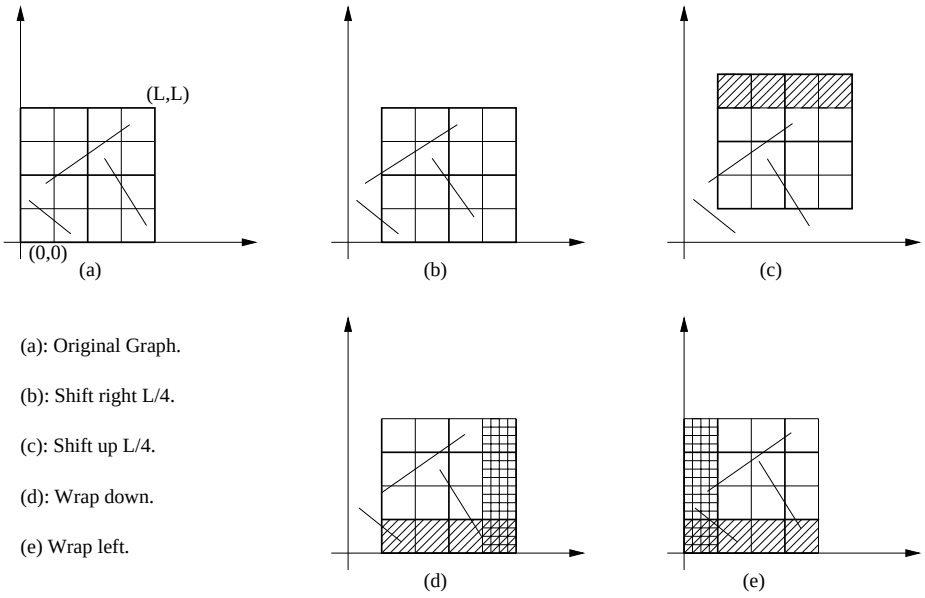


Figure 2: *Dissection with shift* (a, b). *Here* $a = b = \frac{L}{4}$. *Note that the given line segments do not shift. (e) gives the shifted dissection with shift* $(\frac{L}{4}, \frac{L}{4})$.

dissection contains no line segment, it becomes a leaf node.

Remarks: (i) All line segments do not shift; (ii) A "wrapped-around" square is treated as one square. This greatly simplifies the notation latter; (iii) The quadtree with shift (a, b) has a very different structure than the original one. Each square can be an integral square, or a disjoint union of 2 or 4 rectangles; (iv) Shifted dissection can be constructed in polynomial time. A variant of the shifted dissection can be found in [12, 9]; (v) The dissection with shift (a, b) will be used in the proof of Structure Theorem while the quadtree with shift (a, b) will play an important role in dynamic programming to deterministically find the approximate solution to the disturbed instance.

Remarks: (The purpose of perturbation) (i) Let d_{min} be the minimum internode distance. Here nodes refer to segment endpoints and Steiner points. If we want each leaf node of the quadtree holds at most one endpoint or one Steiner point, the quadtree must have depth $O(log\frac{L}{d_{min}})$. For efficiency, we perturb the instance such that the minimum nonzero internode distance is at least 1. This helps to forestall the case when "too close" nodes make the quadtree too big by treating these "too close" nodes as one node. (ii) By perturbation, all endpoints and Steiner points have integral coordinates. Thus, the dissection can be made such that each endpoint and Steiner point is located in the center of some leaf square of the quadtree. This helps to preclude any precision issues which would otherwise arise when the algorithm "guesses" the location of a Steiner point. (iii) We will design an approximate algorithm to the perturbed instance. The output Steiner tree may have "bent" edges since the road edges must cross grid lines at prespecified locations. But

we can smoothen all bent edges after the algorithm is done. This will not increase cost according to triangle inequality.

2.3 Structure Theorem

Let m, r be positive integers and $r \leq m$. Given a dissection with shift (a, b),

Definition 1 (m-regular) *An m-regular set of portals for the shifted dissection is a set of points on the edges of the squares in it. Each square has a portal at each of its 4 corners and $m-1$ other equal-spaced portals on each edge.*

Definition 2 (good Steiner tree) *A good Steiner tree is a valid Steiner tree which connects all input line segments and some subset of portals in the dissection.*

Definition 3 ((m,r)-light) *A good Steiner tree is (m,r)-light with respect to the shifted dissection if (i) its road edges cross each edge of each square in the dissection at most r times and always at a portal; (ii) its segment edges cross each edge of each square in the dissection at most r times (segment edges can not be relocated).*

Remarks: (i) If m is a power of 2, all portals in a square overlap with some portals of its 4 children. (ii) Over the shifted dissection, each line segment is cut into several parts by partition lines. Because each segment has only one point (exit) to connect to the network, only one part has this point. We call this part *primary part* and other parts *secondary parts.* (iii) The segment edge of a (m,r)-light Steiner tree crosses each edge of each square in the quadtree at most r times. How to force this happen with given line segments? An intuitive idea is to remove all secondary parts. This will be explained in Patching Lemma (Lemma 2.5).

Theorem 2.2 (Structure Theorem) *Let OPT be the cost of the optimal Steiner tree for a perturbed instance whose bounding square has size L. Let $c > 1$ be any constant. Let $m = O(c\log(L))$ and $r = O(c)$. Assume the minimum nonzero internode distance in the perturbed instance is at least 4. Pick a shift (a, b) randomly. Then with probability at least $\frac{1}{2}$, there exists a Steiner tree of cost at most $\left(1 + \frac{1}{c}\right)$ OPT that is (m,r)-light with respect to the dissection with shift (a, b).*

Remarks: (i) Structure Theorem guarantees the existence of (m,r)-light Steiner trees. We need to find the optimal one among all Steiner trees which are (m,r)-light with respect to the shifted dissection. This can be done by dynamic programming in polynomial time. (ii) One can compute the (m,r)-light optimal Steiner tree for all choices of (a, b), and then choose the one with smallest cost as the approximation to the problem instance. This derandomization increases the running time by a factor $O(n^2)$ where n is the number of line segments in the given instance.

We will prove the Structure Theorem constructively. We start from an optimal Steiner tree and a randomly shifted dissection, and modify the optimal Steiner tree until it is (m,r)-light with respect to the shifted dissection. The proof uses a top-down fashion with respect

to the shifted dissection tree. Since the optimal Steiner tree may cross some edge of some square in the dissection tree more than r times, How to modify it such that it is (m,r)-light? For this concern we discuss the following truncating and patching strategies.

For any segment edge e of a good Steiner tree τ, if some cut line S crosses e, then one part of e will be in one side of S and the other part of e will be in the other side of S. We call each part of e a *partial segment edge*. Not that if e and S overlap, or one endpoint of e is in S, we think e is located in one side of S which means S and e do not cross.

Lemma 2.3 (Two-way Truncating Lemma) *Let* S *be any line segment and* τ *be a good Steiner tree which crosses S. Then* τ *can be truncated such that no segment edge of* τ *crosses* S.

Proof. We can simply break all crossing line segments at crossing points and remove all partial segment edges which do not connect to any road edge. If the crossing point happens to be an exit, choose either partial segment edge to remove. ■

Lemma 2.3 is called *two-way truncating* since truncation happens in either side of S. Now we give the *one-way truncating* lemma where truncation happens in exactly one side of S. Note that both truncation leave a tree with no segment edge crossing S. Figure 3 illustrates these two kinds of truncations.

Lemma 2.4 (One-way Truncating Lemma) *Let* S *be any line segment with length* s *and* τ *be a good Steiner tree which crosses S. Then there exist line segments on S whose total length is at most* s *and whose addition to* τ *changes* τ *to a good Steiner tree such that all partial segment edges in one side of S are removed tree and all partial segment edges in the other side of S are retained in the tree.*

Proof. Assume τ crosses S $t \geq 1$ times. Let $X = (x_1, x_2, \cdots, x_t)$ be the set of crossing points. Break τ in all these crossings, we get a Steiner forest $\mathcal{F}$. Make two copies of each x_i for each side of S. Denote these copies by x_i' and x_i''. Note that we assume all x_i''s are located in one side of S and all x_i'''s are located in the other side of S. For each x_i, consider each component in $\mathcal{F}$ which connects to x_i in the original graph. If it is located in the same side of S as x_i', connect it to x_i'; Otherwise, connects it to x_i''.

W.l.o.g., we assume the side of S containing all x_i''s has fewer number of components which consist of only partial segment edges. We call this side *side 1* and the other side *side 2*.

(1) Remove all partial segment edges from side 2. If the partial segment in side 2 contains an exit, and the exit connects to only one road edge, remove this road edge.

(2) If both x_i' and x_i'' exist in $\mathcal{F}$, add segment $x_i'x_i''$ to $\mathcal{F}$. Note that the length of $x_i'x_i''$ is 0.

(3) For all x_i''s in side 1, where $i = 1, 2, \cdots, t-1$, if segment $x_i'x_{i+1}'$ connects two disjoint Steiner trees in $\mathcal{F}$, add it to $\mathcal{F}$. The total added segment length is at most s.

Step (1) removes all partial segment edges in side 2. Step (2) is applied if and only if x_i is a crossing point between a road edge and S. In other words, after step (2), $\mathcal{F}$ contains

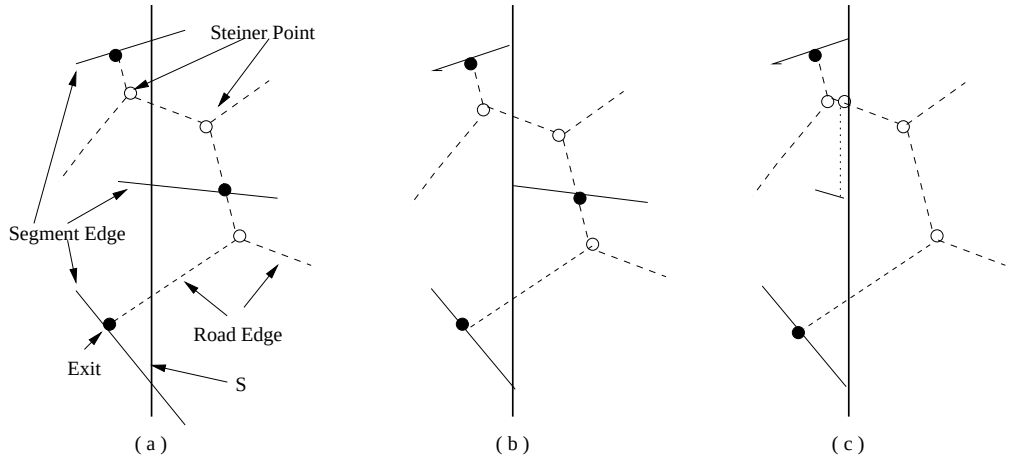


Figure 3: *(a) A partial Steiner tree where solid lines represent segment edges and dashed lines represent road edges. (b) The resulting Steiner tree after two-way truncation. (c) The resulting Steiner tree after one-way truncation. Here the dotted line represents the added segment.*

one Steiner tree containing all existing road edges and some components consisting of only partial segment edges. Then we apply step (3) to connect all these components to the good Steiner tree. Thus the resulting graph is a good Steiner tree with all partial segment edges located in one side of S, and the cost of the tree is increased by at most s. ■

Lemma 2.5 (k-Patching Lemma) *Let* S *be any line segment of length s and τ be a good Steiner tree. If the segment edges of τ cross S more than k times, or the road edges of τ cross S more than k times, or both situation happen, then*
Case 1: If all crossings happen between segment edges and S*, then τ can be modified with no cost increase such that no edge of τ crosses* S*.*
Case 2: If at least one crossing happens between a road edge of τ and S, then there exists line segments on S *whose total length is at most* $2s$ *and whose addition to τ changes τ to a new good Steiner tree such that (i) the road edges of τ cross* S *exactly once; (ii) no segment edges of τ crosses* S*.*

Proof. If all crossings happen between segment edges and S, we can apply the One-way Truncating Lemma to τ. It is easy to see that case 1 holds. Now consider case 2.

If the number of crossings between segment edges of τ and S is $> k$, apply One-way Truncating Lemma; Otherwise, apply Two-way Truncating Lemma. After the truncation, no segment edge of τ crosses S and the the total cost increase is at most s. Let $x_1, x_2, \cdots, x_t$ be the ordered sequence of crossing points along S (in either direction) after the truncation. Note that now all crossings happen between road edges and S. Break τ at these points, we

get a Steiner forest $\mathcal{F}$. Make two copies of each x_i, one for each side of S. Denote these copies by x_i' and x_i''. For all $i = 1, 2, \cdots, t$, let all x_i' be located in one side of S, while all x_i'' be located in the other side of S.

Initially let J be an empty set of line segments. For $i = 1, 2, \cdots, t-1$, if segment $x_i'x_{i+1}'$ (or $x_i''x_{i+1}''$) connects two Steiner trees in $\mathcal{F}$, add $x_i'x_{i+1}'$ (or $x_i''x_{i+1}''$) to J. Finally add $x_t'x_t''$ to J. We claim that $x_i'x_{i+1}'$ and $x_i''x_{i+1}''$ can't connect to two disjoint components in $\mathcal{F}$ at the same time since x_i and x_{i+1} break τ into only 3 disjoint connected components and these 3 components must be located in both sides of S. Thus the total length of all segments in J is at most s since the length of $x_t'x_t''$ is 0.

Add J to $\mathcal{F}$, the resulting Steiner tree has length at most $2s$ more than that of the original Steiner tree and it crosses S exactly once. Case 2 holds. ■

Remarks: (i) Patching Lemma respects the validity of the good Steiner tree. In other words, after patching, each segment still has exactly one point (may not be the original one) connecting to the network. (ii) Patching Lemma tells us that if a Steiner tree crosses an edge of a square in the dissection too many times, we can modify the tree such that it crosses the edge at most once. The addition of line segments lying on the edge introduces "bent" edges to the Steiner tree.

Lemma 2.6 *Let τ be a Steiner tree with cost T whose edge length is at least 4 units. For a grid line l in the unit grid which covers the tree, let $t(\tau, l)$ be the number of times τ crosses l. Then*

$$\sum_{l:\, vertical} t(\tau, l) \quad + \sum_{l:\, horizontal} t(\tau, l) \leq 2 \cdot T$$

Proof. Let e be any edge with length s in τ. Let a and b be the lengths of the horizontal and vertical projections of e. Then e crosses the grid lines at most $a+b+2$ times. We have $a+b \leq \sqrt{2}\, s$ since $a^2+b^2 = s^2$, $ab \leq \frac{s^2}{2}$ and $a+b = \sqrt{s^2 + 2\,a\,b}$. When $s \geq 4$, $\sqrt{2}\, s+2 \leq 2\, s$ which means $a + b + 2 \leq 2\, s$. ■

Remark: Lemma (2.6) relates the length of a Steiner tree to the number of times the tree crosses the grid lines of a unit grid.

The proof of the Structure Theorem uses Markov's inequality. We state it as below.
Markov's inequality: If X is a random variable in $[0, +\infty)$ with the mean μ, then for any positive constant a, $P_r(X \geq a) \leq \frac{\mu}{a}$.

Now we are ready for the proof of Structure Theorem.
Proof of Structure Theorem. Let $s = 10c$, $r = s+2$, $m \geq 2\, s \log L$. Let τ be an optimal Steiner tree which connects all given line segments in a perturbed instance. Suppose shift (a, b) is picked randomly. We will first introduce the concept of *level* for dissection lines, then transform τ into a good Steiner tree which is (m,r)-light with respect to the randomly-shifted dissection. We will charge the cost increase due to transformation to each unit grid line. Note that we assume L is a power of 2. Then the vertical lines and horizontal lines

in the dissection lie in the unit grid lines. From now on, we use grid lines to refer to both dissection lines and unit grid lines.

Recall that the dissection is the recursive partition of the bounding square. And we view a dissection as a hierarchical 4-ary dissection tree. Each node in the dissection tree is one square, and all squares in the same level have equal size. So naturally each square has a level. The root square has level 0. It's 4 child squares have level 1. ... All leaf squares have level $log\, L$. Since the side of each square is part of some vertical or horizontal lines, we introduce a *level* for each grid line accordingly.

In a dissection with shift (a, b), the leftmost vertical line is in x-coordinate a and the lowest horizontal line is in y-coordinate b. These two grid lines have level 0 since they contribute to the level 0 square in the dissection tree; The vertical lines ($x = a$, $x = \frac{L}{2} + a \mod L$) and the horizontal lines ($y = b$, $y = \frac{L}{2} + b \mod L$) have level 1 since they contribute to level 1 squares; Generally speaking, the vertical lines ($x = a + p \cdot \frac{L}{2^i} \mod L$) for $p = 0, 1, \cdots, 2^i - 1$, and horizontal lines ($y = b + p \cdot \frac{L}{2^i} \mod L$) for $p = 0, 1, \cdots, 2^i - 1$, have level i, where $i = 0, i, \cdots, log\, L$, since they contribute to level i squares in the dissection tree. It is clear that a level i grid line is also a level $i + 1, \cdots, log\, L$ grid line. The highest level of a grid line is the highest level among all squares it contributes.

Since the shift (a, b) is picked randomly, for each vertical grid line l and each $1 \leq i \leq log\, L$,

$$P_r[l \text{ is at level } i] = \frac{2^i}{L} \tag{1}$$

$$P_r[\text{the highest level of } l \text{ is } i] = \frac{2^{i-1}}{L} \tag{2}$$

(Because totally there are L vertical lines, 2^i of them are at level i and 2^{i-1} of them have highest level i.) Similar arguments hold for horizontal lines.

Now we want to modify the optimal Steiner tree τ into a good Steiner tree which is (m,r)-light. We first make τ (m,s)-light. Then we explain why we need the 2 extra crossings on each square edge in the shifted dissection tree.

The modification will be done in a topdown fashion with respect to the dissection tree with random shift (a, b). At level i, there are 4^i squares. We will first make both road edges and segment edges of τ cross each edge of each level i square at most s times; Then we force the crossings with road edges of τ happen at portals. When all levels are visited, we get a (m,r)-light Steiner tree. We will use probability method (over random shift (a, b)) to analyze this procedure.

How to modify τ such that both road edges and segment edges cross each edge of each level i square at most s times? the relationship between grid lines with highest level i and level i squares suggests the following way: Go through all grid lines. If a grid line l has highest level i, then go through each of its 2^i segments of length $\frac{L}{2^i}$ and applying s-Patching Lemma whenever one of them is crossed by the road edges or segment edges of the current good Steiner tree more than s times. But we can not apply Patching Lemma directly to the segment with $> s$ crossings because we may introduce too much cost increase. Since l is also at level $i + 1, \cdots, log\, L$, a better idea is to apply Patching Lemma bottom up for all

$i \leq j \leq log\, L$. The following procedure MODIFY(l, i, a) does the patching for vertical line l whose highest level is i in a dissection with random horizontal shift a.

MODIFY(l, i, a)
  For $j = log\, L$ down to i do
    For $p = 0, 1, \cdots, 2^{j-1}$, if the segment of l between the y-coordinates $(a + p \cdot \frac{L}{2^j} \bmod L)$ and $(a + (p+1) \cdot \frac{L}{2^j} \bmod L)$ is crossed by the road edges or segment edges of the current good Steiner tree more than s times, then apply s-Patching Lemma to reduce the number of crossings to 2.

Remarks on MODIFY: (i) Patching Lemma reduces the number of crossings to 2 instead of 1 because we must do patching separately for the 2 parts of a "wrapped around" square. (ii) The modifications are done in the "bottom-up" fashion. When we consider squares in level $i+1, \cdots, log\, L$ in the dissection tree, we do not need to check l again even though l contributes to some squares in level $i+1, \cdots, log\, L$. In other words, all charges to grid line l whose highest level is i are due to the level i modification. (iii) The patching increases the cost of the current good Steiner tree. It may also increase the number of times the good Steiner tree crosses a horizontal line h. If the patching along l increases the number of crossings of the tree with h at least twice, we can apply 1-Patching Lemma to these crossings and reduce the number to 2 (do patching for each side of l). This patching does not increase cost since all crossings happen in the cross point of l and h. That's the reason why we choose $r = s + 2$ to make the good Steiner tree (m,r)-light. (iv) A similar procedure and arguments hold for horizontal lines with shift b.

Let $c_{l,j}(a)$ be the number of segments to which we do patching for each j in MODIFY(l, i, b). Note that $c_{l,j}(a)$ is independent of i since the loop computation for j does not depend on i in MODIFY. Thus

$$\sum_{j \geq 1} c_{l,j}(a) \leq \frac{t(\tau, l)}{s-1} \tag{3}$$

where $t(\tau, l)$ denote the number of times the good Steiner tree τ crosses grid line l. The inequality holds because each application of Patching Lemma counted on the lefthand side replaces at least $s+1$ crossings by at most 2. Thus the cost increase due to patching in MODIFY is:

$$\Delta cost_{patching}(l, i, a) \leq \sum_{j \geq i} c_{l,j}(a) \cdot \frac{2L}{2^j} \tag{4}$$

where the term $\frac{2L}{2^j}$ comes from Patching Lemma. We charge this cost to l. Of course only when the highest level of l is i do we charge l the cost increase due to patching.

Similar analysis holds true for level i horizontal lines.

Now in the current good Steiner tree both road edges and segment edges cross each edge of each level i square in the shifted dissection tree at most m times. We need to move each

crossing point with road edges to its nearest portal. We call this procedure *relocation*. Note that crossing points with segment edges do not relocate.

If a line l is at level i, each of the $t(\tau, l)$ crossings might have to be displaced by $\frac{L}{2^{i+1}\, m}$ to reach its nearest portal. Instead of actually deflecting the edge, we break it at the crossing point and add two line segments (one on each side of l) of total length at most $\frac{L}{2^i\, m}$ such that the new edge crosses l at a portal. Thus we have

$$\Delta cost_{relocation}(l, i, a) \leq t(\tau, l) \cdot \frac{L}{2^i\, m} \tag{5}$$

From the above analysis, we know that the total cost increase charged to a level i vertical line l is:

$$\Delta cost_{total}(l, i, a) \leq t(\tau, l) \cdot \frac{L}{2^i\, m} + \sum_{j \geq i} c_{l,j}(a) \cdot \frac{2L}{2^j} \tag{6}$$

while to a level i horizontal line l is:

$$\Delta cost_{total}(l, i, b) \leq t(\tau, l) \cdot \frac{L}{2^i\, m} + \sum_{j \geq i} c_{l,j}(b) \cdot \frac{2L}{2^j} \tag{7}$$

Consider all vertical grid lines. With probability $\frac{2^{i-1}}{L}$ over random shift (a, b) (see (2)), a vertical grid line has highest level i. According to the Linearity of Expectations, the expected cost increase charged to a vertical grid line l is:

$$\begin{array}{rcl} E_a[cost(l, i, a)] & \leq & \sum_{i=1}^{log\, L} \frac{2^{i-1}}{L} \cdot \left[t(\tau, l) \cdot \frac{L}{2^i\, m} + \sum_{j \geq i} c_{l,j}(a) \cdot \frac{2L}{2^j} \right] \\ & = & \frac{log\, L}{2\, m} \cdot t(\tau, l) + \sum_{i=1}^{log\, L} 2^i \sum_{j \geq i} \frac{1}{2^j} \cdot c_{l,j}(a) \\ & = & \frac{log\, L}{2\, m} \cdot t(\tau, l) + \sum_{j=1}^{log\, L} \frac{c_{l,j}(a)}{2^j} \sum_{i=1}^{j} 2^i \\ & \leq & \frac{log\, L}{2\, m} \cdot t(\tau, l) + 2 \cdot \sum_{j=1}^{log\, L} c_{l,j}(a) \\ & \leq & \frac{log\, L}{2\, m} \cdot t(\tau, l) + \frac{2\, t(\tau, l)}{s - 1} \\ & \leq & \frac{5\, t(\tau, l)}{2\, s} \end{array} \tag{8}$$

where the last inequality holds when $s \geq 9$ and $m \geq 2\, s\, log\, L$. Similarly for any horizontal line l, and horizontal shift b, we have

$$E_b[cost(l, i, b)] \leq \frac{5\, t(\tau, l)}{2\, s} \tag{9}$$

From Lemma 2.6 and again Linearity of Expectations, the expected total cost increase due to modification is:

$$\sum_{l:\, vertical} \frac{5\, t(\tau, l)}{2\, s} + \sum_{l:\, horizontal} \frac{5\, t(\tau, l)}{2\, s} \leq \frac{5\, OPT}{s} \tag{10}$$

Since $s = 10c$, the expected cost increase is at most $\frac{OPT}{2\, c}$. According to Markov's Inequality, with probability at least $\frac{1}{2}$ the cost increase is no more than $\frac{OPT}{c}$. ■

2.4 The Algorithm

Structure Theorem tells us that with probability at least $\frac{1}{2}$, There exists a (m,r)-light Steiner tree with cost at most $(1+\frac{1}{c})OPT$. This suggests the following algorithm:

Step 1. Perturb and rescale the given instance.

Step 2. Construct a quadtree with random shift (a, b) based on the perturbed instance.

Step 3. Find an optimal (m,r)-light Steiner tree with respect to the dissection with shift (a, b) by dynamic programming.

We have explained step 1 and 2 in detail in previous sections. Now we focus on the dynamic programming procedure.

The basic idea comes from the following observation. Suppose S is a square in the shifted quadtree and the optimal (m,r)-light Steiner tree crosses the boundary of S a total of $\leq 4\,r$ times. Not that all the crossings with road edges happen at portals.

The part of the optimal (m,r)-light Steiner tree residing inside S forms a *Steiner forest* of p trees with the following characteristics: (i) each of the p trees connects some line segments and some portals of S. (ii) all p trees together connect all line segments completely inside S and a subset of line segments which cross S. (iii) the collection of these p trees is (m,r)-light. That is, they collectively cross each edge of each square in the subtree rooted at S at most r times, and the crossings with road edges always happen at portals.

Since the (m,r)-light Steiner tree is optimal, each of the p trees must be optimal. This motivates us to define the following (m,r)-light Steiner forest problem. An instance of the (m,r)-light Steiner forest problem is specified by

(a) A square S in the shifted quadtree.
(b) A multiset B containing $\leq r$ portals on each of the four sides of S.
(c) A partition of B into $(B_1, B_2, \cdots, B_p)$.
(d) A subset M containing $\leq 4r$ line segments which cross S. The number of crossings on each of the four sides of S is $\leq r$.

Note that if one edge (denoted by $s1$) of S crosses more than m line segments, we break all these line segments at the crossings and choose to let S include all these segments by removing all partial line segments out of S, or to totally omit all these line segments by removing all partial line segments inside S. In the first case, all these crossing line segments are completely inside S. In the second case, none of them is inside S. In both cases, M contains none of these line segments. If the number of crossings is $\leq m$, we choose a subset (with cardinality $\leq r$) of these line segments and include the subset in M. If one line segment crosses two edges of S, the decision must respect the singularity of the line segment, that is, either S contains the line segment, or S does not contain it at all.

We are going to find an optimal (m,r)-light collection of p Steiner trees with the ith tree contains all portals in B_i and all trees together connect all line segments which are

completely inside S and all line segments in M which cross S. Note that the crossing points with line segments in M are located in S. Each instance in S can be broken into many smaller and simpler instances of (m,r)-light Steiner forest problem in the 4 child squares of S. Instances corresponding to the leaf nodes of the quadtree are solved in a brute-force way.

We build a lookup table in a bottom-up fashion for dynamic programming. The number of entries in the table equals

$$\sharp(a) \times \sharp(b) \times \sharp(c) \times \sharp(d)$$

where $\sharp(x)$ is the number of choices for (x).

We already know that $\sharp(a) = O(n^2 \, log \, (n))$. With a simple combinatorial analysis, we have $\sharp(b) = O(2^{4m+4r})$, $\sharp(c) = O(2^{O(4r)})$ (no edge crossings), and $\sharp(d) = O(2^{4m})$. Thus the total number of entries in the lookup table is $O(n^2 \, log \, (n) \cdot 2^{8m+O(4r)})$. When $m = O(\frac{log \, n}{c})$, this number becomes $O(n^{O(c)} log(n))$.

If an entry corresponds to an instance in a leaf node of the quadtree, since there are at most $O(r)$ portals and at most one line segment, we can solve it by brute-force which takes $O(2^{O(r)})$ time. If an entry corresponds to an instance in the root node of the quadtree, no portals will be chosen for multiset B and no crossing line segments will be chosen for M since the root node contains all line segments. The result is an (m,r)-light Steiner tree and is the output of the dynamic programming. For all other entries, we need to compute the (m,r)-light Steiner forest inductively. Let S be any nonleaf square in the quadtree and let S_1, S_2, S_3, S_4 be its four children. Each choice of (b), (c), (d) for S decides an instance I. To compute the entry for I, we must consider all possible ways in which an (m,r)-light Steiner tree could cross the edges of S_1, S_2, S_3, S_4. This means we need to enumerate all triples of the following kind:

(a') A multiset B' containing $\leq r$ portals in each of the 4 inner edges of the child squares. (The outer edges of the 4 children are part of the edges of S. So, the number of portals, the partition of these portals, and the crossing line segments are all set.)

(b') A partition of B' into $(B_1, B_2, \cdots, B'_{p'})$.

(c') A subset M'_i for each of the 4 child squares containing $\leq r$ line segments crossing each of the two inner edges of the child square, where $i = 1, 2, 3, 4$.

When we choose M'_i for each child square, we must consider the following situations since each inner edge contributes to two squares. Let's use squares S_1 and S_2 which share the common inner edge e as an example. If the number of line segments crossing e is $> m$, We have only two choices. We either let S_1 contains all of these line segments by removing all partial segments out of S_1, or let S_2 contains all of these line segments by removing all partial segments out of S_2. In both cases, M'_1 and M'_2 contain none of these line segments. If the number of line segments crossing e is $\leq m$, for each of these line segments, we choose to let either M'_1 or M'_2 contains this segment.

Note that each choice of (a'), (b') and (c') decides a smaller instance for each of the 4 child squares. Since the optimal (m,r)-light Steiner forest to all instances in the 4 child squares are already in the lookup table (by induction), we do not need to compute them again. We union the 4 (m,r)-light Steiner forests to get an (m,r)-light Steiner forest for instance I

but this Steiner forest may not be optimal. Each choice of (a'), (b') and (c') results in an (m,r)-light Steiner forest for I and the one with smallest cost is the one that will be put into the table for instance I. Since $\sharp(a') = O(2^{4m+4r})$, $\sharp(b') = O((4r)!\, 2^{O(4r)})$(no edge crossings) and $\sharp(c') = O(2^{4m})$ the running time for each entry takes $O((4r)! \cdot 2^{8m+O(4r)}) = O(n^{O(c)})$ when $m = O(\frac{log(n)}{c})$ and $r = O(c)$.

For those "wrapped-around" squares, we treat them as normal squares, except that they are already divided into 2 or 4 rectangles. The choices of (b), (c) and (d) and the choices of the corresponding (a'), (b') and (c') must respect these divisions.

Remarks on dynamic programming: (i) The dynamic programming procedure described above computes only the cost of the optimal (m,r)-light Steiner tree. We can construct the (m,r)-light Steiner tree by looking at the decisions made for each step of the dynamic programming. (ii) As mentioned earlier, We can run dynamic programming for all choices of random numbers a and b, and choose the output with smallest cost for the given instance. This increase the running time by a factor $O(n^2)$. (iii) In the induction analysis of dynamic programming for instance I, the portals in the 4 edges of S may not overlap with portals in the outer edges of the 4 children of S. Thus we can not union the optimal solutions to the 4 children to get a (m,r)-light Steiner forest for I. This problem can be avoided by forcing m to be a power of 2. (iv) Steiner nodes may be introduced when we compute the (m,r)-light Steiner forest for each leaf node of the quadtree.

3 Conclusion and Discussion

In this paper we study the problem of Interconnecting Highways. This is the generalized minimum Steiner Tree problem where the terminals are located in some line segments. We not only need to "guess" the locations of Steiner points, but also need to locate the terminal positions, called exits, in each line segment. We present a PTAS for this problem.

Some similar problems arise in these years. For example, given n fixed nonempty convex polytopes in the Euclidean d-space R^d, find a shortest network to connect all these polytopes with the constraints that each polytope has exactly one point to be connected to the network [14]. Another example, given a set of local networks, construct a global network to connect all the local networks. The global network must form a connected network of its own [11]. These two problems can be treated as the generalized Interconnecting Highways problem. And polynomial time approximation schemes may exist by including some parameters in the objective functions and applying similar idea.

References

[1] S. Arora, Polynomial Time Approximation Schemes for Euclidean Traveling Salesman and other Geometric Problems, *Proceedings of 37th IEEE Symposium on Foundations of Computer Science*, (1996) 12

[2] S. Arora, Polynomial Time Approximation Schemes for Euclidean TSP and other Geometric Problems, (1997). Available from *http:www.cs.princeton.edu/~arora*

[3] M. Bern, D. Eppstein and S.-H. Teng, Parallel Construction of Quadtree and Quality Triangulation, *Proc. 3rd WADS*, Springer Verlag *LNCS* 709, (1993) 188-199

[4] G.X. Chen, The shortest path between two points with a (linear) constraint, *Knowledge and Appl. of Math.*, 4 (1980) 1-8

[5] D.-Z. Du, F.K. Hwang, A Proof of Gilbert-Pollak's conjecture on the Steiner ratio, *Algorithmica*, 7 (1992) 121-135

[6] D.-Z. Du, F.K. Hwang, and G.L. Xue, Interconnecting Highways, *SIAM Discrete Mathematics*, 12 (1999) 252-261

[7] D.-Z. Du, B. Lu, H. Ngo and P.M. Pardalos, Steiner Tree Problems, *manuscript* (2000)

[8] M.R. Garey, R.L. Graham and D.S. Johnson, The complexity of computing Steiner minimal trees, *SIAM J. Appl. Math.*, 32 (1977) 835-859

[9] D.S. Hochbaum and W. Maass, Approximation Schemes for Covering and Packing Problems in Image Processing and VLSI, *J. ACM*, 32 (1985) 130-136

[10] F.K. Hwang, D.S. Richards and P. Winter, The Steiner Tree Problem, *Annals of Discrete Mathematics*, North-Holland, (1992)

[11] E. Ihler, G. Reich and P. Widmayer, On Shortest Networks for Classes of Points in the Plane, *LNCS* 553 (1991) 103-111

[12] T. Jiang and L. Wang, An Approximation Scheme for some Steiner Tree Problems in the Plane, *LNCS* 834 (1994) 414-422

[13] J.F. Weng, Generalized Steiner Problem and Hexagonal Coordinate (in Chinese), *Acta Math. Appl. Sinica*, 8 (1985), 383-397

[14] G.L. Xue, D.-Z. Du, and F.K. Hwang, Faster Algorithm for Shortest Network under Given Topology, *manuscript*