# Approximating Minimum Manhattan Networks
(Extended Abstract)

Joachim Gudmundsson[1] *, Christos Levcopoulos[1] **, and Giri Narasimhan[2] ***

[1] Department of Computer Science, Lund University, Box 118, 221 00 Lund, Sweden
[2] Department of Mathematical Sciences, The University of Memphis, Memphis, TN 38152, USA

**Abstract.** Given a set $S$ of $n$ points in the plane, we define a *Manhattan Network* on $S$ as a rectilinear network $G$ with the property that for every pair of points in $S$, the network $G$ contains the shortest rectilinear path between them. A *Minimum Manhattan Network* on $S$ is a Manhattan network of minimum possible length. A Manhattan network can be thought of as a graph $G = (V, E)$, where the vertex set $V$ corresponds to points from $S$ and a set of steiner points $S'$, and the edges in $E$ correspond to horizontal or vertical line segments connecting points in $S \cup S'$. A Manhattan network can also be thought of as a 1-spanner (for the $L_1$-metric) for the points in $S$.

Let $R$ be an algorithm that produces a rectangulation of a staircase polygon in time $R(n)$ of weight at most $r$ times the optimal. We design an $O(n \log n + R(n))$ time algorithm which, given a set $S$ of $n$ points in the plane, produces a Manhattan network on $S$ with total weight at most $4r$ times that of a minimum Manhattan network. Using known rectangulation algorithms, this gives us an $O(n^3)$-time algorithm with approximation factor four, and an $O(n \log n)$-time algorithm with approximation factor eight.

## 1 Introduction

A *rectilinear path* connecting two points in the plane is a path consisting of only horizontal and vertical line segments. A shortest rectilinear path connecting two points will be referred to as a *Manhattan path*. Manhattan paths are monotonic. The *Manhattan distance* (or $L_1$-distance) between two points in the plane is the length of the Manhattan path connecting them. In this paper we introduce the concept of geometric networks that guarantee Manhattan paths between every pair of points from a given set of points.

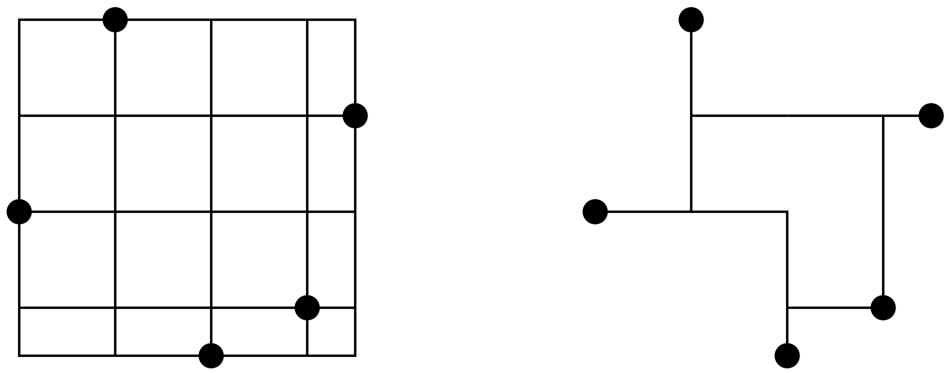
Consider a set $S$ of points in the plane. A *geometric network* on $S$ can be modeled as an undirected graph $G = (V, E)$. The vertex set $V$ corresponds to the points in $S \cup S'$, where $S'$ is a set of newly added Steiner points; the edge set $E$ corresponds to line segments joining points in $S \cup S'$. If all the line segments are either horizontal or vertical, then the network is called a *rectilinear geometric network*. Each edge $e = (a, b) \in E$ has length $|e|$ that is defined as the Euclidean distance $|ab|$ between its two endpoints $a$ and $b$. The total length of a set of edges is simply the sum of the lengths of the edges in that set. The *total length* of a network $G(V, E)$ is denoted by $|E|$. For $p, q \in S$, a $pq$-path in $G$ is a path in $G$ between $p$ and $q$.

For a given set $S$ of $n$ points in the plane, we define a *Manhattan Network* on $S$ as a rectilinear geometric network $G$ with the property that for every pair of points $p, q \in S$, the network $G$ contains a Manhattan $pq$-path connecting them. A *Minimum Manhattan Network* on $S$ is a Manhattan network of minimum possible length.

The complete grid on the point set $S$ is clearly a Manhattan network. In other words, the network obtained by drawing a horizontal line and a vertical line through every point in $S$ and by considering only the portion of the grid inside the bounding box of $S$ is a network that includes the

* email: `Joakim.Gudmundsson@dna.lth.se`
** email: `Christos.Levcopoulos@dna.lth.se`
*** email: `giri@msci.memphis.edu`; Funded by grants from NSF (CCR-940-9752) and Cadence Design Systems, Inc.

Manhattan path between every pair of points in $S$. It is easy to show that the minimum Manhattan network on $S$ need not be unique and that the complete grid on the point set $S$ can have total weight $O(n)$ times that of a minimum Manhattan network on the same point set. Figures 1(a)



(a) (b)

**Fig. 1.** (a) A Manhattan network, and (b) A minimum Manhattan network.

and (b) below show examples of a Manhattan network and a minimum Manhattan network on the same set of points. In fact, the network in Figure 1(a) is also a complete grid on the input points. The concept of Manhattan networks seems to be a very natural concept; it is surprising that this concept has not been previously studied. Manhattan networks are likely to have many applications in geometric network design and in the design of VLSI circuits.

Manhattan networks are also closely connected to the concept of spanners. Given a set $S$ of $n$ points in the plane, and a real number $t>1$, we say that a geometric network $G$ is a *t-spanner* for $S$ for the $L_p$-norm, if for each pair of points $p, q \in S$, there exists a $pq$-path in $G$ of length at most $t$ times the $L_p$-distance between $p$ and $q$. In this connection, a minimum Manhattan network can be thought of as a sparsest 1-spanner for $S$ for the $L_1$-norm, assuming that Steiner points are allowed to be added. Note that the sparsest 1-spanner for $S$ in the $L_2$-norm is simply the complete graph on $S$. It is also interesting to note that a Manhattan network can be thought of as a $\sqrt{2}$-spanner (with Steiner points) for the $L_2$ norm. Although complete graphs represent ideal communication networks, they are expensive to build; sparse spanners represent low cost alternatives. The weight of the spanner network is a measure of its sparseness; other sparseness measures include the number of edges, maximum degree and the number of steiner points. Spanners have applications in network design, robotics, distributed algorithms, and many other areas, and have been a subject of considerable research [1, 3–6]. More recently, spanners have found applications in the design of approximation algorithms for problems such as the traveling salesperson problem [2, 9].

In this paper we present an algorithm that produces a Manhattan network for a given set $S$ of $n$ points in the plane. The total weight of the network output by the algorithm is within a constant factor of the minimum Manhattan network. It is interesting to note that in this paper we reduce the problem of computing an approximate minimum Manhattan network to the problem of finding a minimum-weight rectangulation of a set of staircase polygons. If the rectangulation algorithm runs in time $O(R(n))$ and produces a rectangulation that is within a factor $r$ of the optimal, our algorithm will produce a Manhattan network of total weight $4r$ times the weight of a minimum

Manhattan network in time $O(n \log n + R(n))$. Using two known approximation algorithms for the minimum-weight rectangulation problem, we obtain two algorithms for the approximate minimum Manhattan network problem. The first algorithm runs in $O(n^3)$ time and produces a Manhattan network of total weight at most four times that of a minimum Manhattan network. The second algorithm runs in $O(n \log n)$ time and has an approximation factor of eight. It is unknown whether the problem of computing the minimum Manhattan network is a NP-hard problem. It is also unknown whether a polynomial-time approximation scheme exists for this problem.

In Section 3, we present the approximation algorithm. In Section 3.2 we prove that the algorithm produces a Manhattan network; in Section 3.3 we prove that the network produced is of weight at most $4r \times |E_{opt}|$, where $E_{opt}$ is the set of edges in a minimum Manhattan network on $S$.

## 2 Definitions

Let $u$ and $v$ be two points in $S$, where $u$ lies to the left of and above $v$. A $\llcorner$-path between $u$ and $v$ is a rectilinear path consisting of one vertical line segment with upper endpoint at $u$, and one horizontal line segment with right endpoint at $v$. Note that such a path is a minimum-weight, minimum-link path connecting $u$ and $v$. The $\urcorner$-path is defined in a similar fashion, as shown in Fig. 2. If $u$ lies to the left of and below $v$, we define a $\lrcorner$-path and a $\ulcorner$-path in a similar fashion. Note that each of the four paths described above introduces one steiner point at the bend. We will denote by $[v, u]$ the closed region described by a rectilinear rectangle with corners in $v$ and $u$. The coordinates of a vertex $v \in V$ are denoted $v.x$ respectively $v.y$.

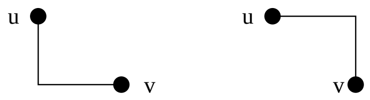



**Fig. 2.** (left) An $\llcorner$-path, and (right) an $\urcorner$-path.

If two different edges of the graph intersect, we will assume that their intersection defines a Steiner point.

## 3 The approximation algorithm

In this section we present an approximation algorithm to construct a small length Manhattan network $G'=(V', E')$. The algorithm will construct the edge set in four independent steps. In each step, for each vertex, the algorithm constructs a (possibly empty) local network connecting that vertex to a set of chosen "neighboring" vertices. Since the four steps are symmetrical the first step is explained in more detail than the later steps.

1. Sweep the points from left to right. As shown in Fig. 3(a), for each point $v' \in V$ let $v$ be the leftmost point below and not to the left of $v'$ (if several, take uppermost). We say that $v'$ *1-belongs to* $v$. For each vertex $v$ let $B_1(v)$ denote the set of vertices in $V$ that 1-belong to $v$. Let $v_1, \ldots, v_m$ be the vertices in $B_1(v)$ ordered from left to right, as shown in Fig. 3(b). First, construct an $\urcorner$-path, denoted by $e_1$, connecting $v_1$ and $v$. If $m>1$, draw a vertical edge $e_2$, with top endpoint at $v_m$ and bottom endpoint on $e_1$. Now, we construct a "local" Manhattan network such that there is a Manhattan path from each vertex $v_i$, $2 \leq i \leq v_{m-1}$, to $e_1$ or $e_2$; this step is

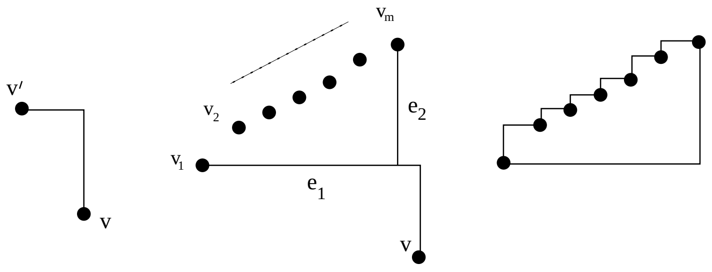



**Fig. 3.**

explained in more detail in Section 3.1. Add the edges constructed in this step to the set of edges $E'_1$.

2. Sweep the points from left to right. For each point $v' \in V$ let $v$ be the leftmost point above and not to the left of $v'$ (if several, take bottommost). We say that $v'$ *2-belongs to* $v$. For each vertex $v$ let $B_2(v)$ denote the set of vertices in $V$ that 2-belong to $v$. Perform the symmetrical procedure as performed in step 1 on the set $B_2(v)$ for every $v \in V$, to obtain the set of edges $E'_2$.
3. Sweep the points from bottom to top. For each point $v' \in V$ let $v$ be the bottommost point to the left and not below $v'$ (if several, take leftmost). We say that $v'$ *3-belongs to* $v$. For each vertex $v$ let $B_3(v)$ denote the set of vertices in $V$ that 3-belong to $v$. Perform the symmetrical procedure as performed in step 1 on the set $B_3(v)$ for every $v \in V$, to obtain the set of edges $E'_3$.
4. Sweep the points from top to bottom. For each point $v' \in V$ let $v$ be the topmost point to the left and not above $v'$ (if several, take leftmost). We say that $v'$ *4-belongs to* $v$. For each vertex $v$ let $B_4(v)$ denote the set of vertices in $V$ that 4-belong to $v$. Perform the symmetrical procedure as performed in step 1 on the set $B_4(v)$ for every $v \in V$, to obtain the set of edges $E'_4$.

After building the appropriate "local" networks, we say that every vertex $v$ is *directly connected* to the vertices in $B_1(v) \cup \ldots \cup B_4(v)$. From these four sweeps we get four edge sets, $E'_1, \ldots, E'_4$. Now the Manhattan network is defined as $G = (V', E')$, where $E' = E'_1 \cup \ldots \cup E'_4$, and $V'$ includes the points in $S$ and all the Steiner points that are generated when adding the edges in $E'$.

Also note that there is some asymmetry in the above construction. This is deliberate; the asymmetric cases are required in the proof of correctness of the algorithm (see Lemma 3).

### 3.1 Constructing the local networks

We will explain the construction of the local networks involved in step 1 of the algorithm in detail. The constructions involved in the other steps are symmetrical. Let $v$ be an arbitrary vertex of $V$ and let $v_1, \ldots, v_m$ be the vertices in $B_1(v)$. By step 1 of the algorithm, we note that $v_i$ lies below and to the left of $v_{i+1}$, $1 \le i < m$; thus, $v.x \ge v_m.x$ and $v.y \le v_1.y$.

In this section we describe how to construct a local network connecting $v$ with the vertices $v_1, \ldots, v_m$; the local network is a Manhattan network on $v, v_1, \ldots, v_m$. We assume that $m > 2$, otherwise we are done. Recall that $v_1$ and $v$ are connected by an $\urcorner$-path, $e_1$, and that $v_m$ is connected to this edge by a vertical segment corresponding to edge $e_2$, as shown in Fig. 3(b). Let $e'_1$ be the horizontal part of $e_1$ between $v_1$ and $e_2$. Our aim is to produce a set of edges of minimum total weight such that there is a Manhattan path between $v_i$ and $v$, $1 \le i \le m$. This already holds for $v_1$ and $v_m$. Consider the following staircase polygon obtained by adding $\ulcorner$-paths between $v_i$ and $v_{i+1}$,

$1 \leq i < m$, to the base $e_1'$ and the right side $e_2$, as shown in Fig. 3(c). This polygon will be referred to as the $C$-hull of the set of vertices $\{v_1, \ldots, v_m\}$. We claim that a rectangulation of this polygon would give us a rectilinear network that guarantees Manhattan paths from $v$ to every vertex in the set $\{v_1, \ldots, v_m\}$. This claim is easily proved by observing that every vertex $v_i, i = 1, \ldots, m$, must lie on the corner of a disinct rectangle (hence you should be able to proceed either down or to the right from $v_i$), and there always exists a monotonic rectilinear path from $v_i$ to $v$ that follows the borders of the rectangles encountered along the way. Denote by $E_1'(v)$ the set of edges constructed in the rectangulation of the $C$-hull plus the two edges $e_1$ and $e_2$. That is, $E_1'(v)$ is the set of edges produced in step 1 to connect the vertices in $B_1(v)$ to $v$. It should be noted that $E_1'(v)$ only consists of the edges in the interior of the $C$-hull of the set of vertices $\{v_1, \ldots, v_m\}$ and that the $\ulcorner$-paths from $v_i$ to $v_{i+1}$ are not included in it. While a rectangulation of these $C$-hulls guarantees Manhattan paths, we show later that a minimum-weight rectangulation of the $C$-hulls gives us an approximation algorithm for the problem. The following results on minimum-weight rectangulations were previously known:

**Theorem 1.** *([8]) An optimal rectangulation of a staircase polygon can be computed in time $O(n^3)$.*

**Theorem 2.** *([7]) A thickest-first rectangulation of a staircase polygon can be computed in linear time such that the weight of the added edges in the rectangulation is at most twice the weight of an optimal rectangulation.*

Since a sweep takes $O(n \log n)$, if the optimal rectangulation procedure is used the time-complexity of our approximation algorithm is $O(n^3)$, and if the thickest-first rectangulation algorithm is used then our algorithm runs in time $O(n \log n)$.

It remains to prove that the graph $G'=(V', E')$ is a Manhattan network for the points in $S$ and that $|E'| \leq 4r \times |E_{opt}|$, where $r$ is the approximation factor of the rectangulation algorithm.

### 3.2 The Algorithm outputs a Manhattan network

To show that the algorithm outputs a Manhattan network, it suffices to prove the following lemma:

**Lemma 3** *For each pair of points $s, t \in V$ there is a Manhattan path in $G'$ connecting $s$ and $t$.*

*Proof.* Without loss of generality, either $t$ lies to the right and above $s$, or $t$ lies to the right and below $s$ (if not, switch $s$ and $t$). We first assume that $t$ lies to the right and above $s$. Without loss of generality, we may assume that no two points have the same $x$- or $y$-coordinate, since the algorithm always connects two such points by a line segment.

Consider $[s, t]$, the rectangle with corners at $s$ and $t$. If $s \in B_2(t)$ or $t \in B_4(s)$, then $s$ is directly connected to $t$ and we are done. Otherwise, we know from the algorithm that $s$ is directly connected to a point $s_1 \in V$ above and to the right of $s$ and to the left of $t$ (Case 4 of construction), and that $t$ is directly connected to a point $t_1 \in V$ below and to the left of $t$ and above $s$ (Case 2 of construction). We consider the following two cases:
**Case 1** [$s_1$ or $t_1$ lies within $[s, t]$]: Without loss of generality, assume that $s_1$ lies within $[s, t]$, as shown in Fig. 4(a). In this case, we let $s_1$ be the new $s$ and continue recursively.
**Case 2** [$s_1$ and $t_1$ lie outside $[s, t]$]: Then we know that $s_1$ lies above and to the left of $t$ and that $t_1$ lie above and to the left of $s$, as shown in Fig. 4(b). The Manhattan path connecting $s$ and $s_1$ must intersect the Manhattan path connecting $t$ and $t_1$ within $[s, t]$. Hence there is a Manhattan path connecting $s$ and $t$.

The case when $t$ lies to the right and below $s$ uses Cases 1 and 3 of the construction, and is otherwise similar to the proof above. Hence the lemma. □

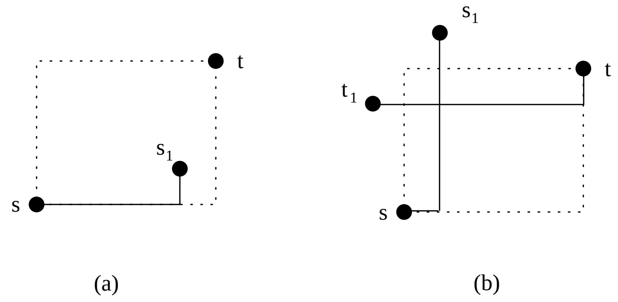



**Fig. 4.**

### 3.3 The total length of the output network is $\leq 4r \times |E_{opt}|$

For the length analysis, once again, it is sufficient to consider only one sweep, i.e., step 1 of the algorithm. The approximation factor for this sweep is then multiplied by four to obtain the approximation factor for the algorithm. Let $v$ be an arbitrary vertex of $V$. Let $B_1(v) = \{v_1, \ldots, v_m\}$, let $V' = B_1(v) \cup \{v\}$. We define a *charging area* of $v$ (with respect to this sweep) as the region $\cup_{i=1}^{m}[v, v_i]$, as shown by the shaded region in Fig. 5(a). The charging area is denoted by $C_1(v)$. Note that the interior of the charging area for any vertex must be devoid of input points. We start with the following observation:

**Lemma 4** *For every pair of vertices $v_i, v_j \in V$, the charging areas $C_1(v_i)$ and $C_1(v_j)$ are disjoint, except possibly for the point $v_i$ or $v_j$.*

*Proof.* Since no vertex can 1-belong to more than one vertex, the staircase parts of the two charging areas cannot share any vertices. Thus either $v_i$ is on the staircase part of $C_1(v_j)$ or vice versa. But then, the rest of the charging areas cannot overlap because of its shape and orientation. □

It is important to point out that the charging areas may share a point, but cannot share an edge of the boundary. Also note that all edges of $E'_1(v)$ that were added in step 1 must lie entirely within $C_1(v)$. Hence, the edges produced in step 1 connecting vertices in $V'$ cannot be used to connect vertices in any other charging areas. We will prove that $|E'_1(v)|$ is at most equal to the length of the edges in $E_{opt}$ lying within the charging area of $v$. Since the charging areas are disjoint this implies that the total length of the edges produced in step 1 of the algorithm, $|E'_1| = \sum_{v \in V} |E'_1(v)|$, is at most $|E_{opt}|$.

First we partition the charging area $C_1(v)$ into three regions, where $R_1 = [v, v_1]$, $R_2 = [v, v_m] \setminus R_1$ and $R_3$ is the remaining region of the charging area. The three regions are shown in Fig. 5(b). Before we continue, we recall that $E'_1(v)$ consists of a $\urcorner$-path, $e_1$, connecting $v_1$ and $v$, a vertical edge $e_2$ connecting $v_m$ with $e_1$, and a rectangulation of the $C$-hull of $B_1(v)$ (which is meant to connect $v_i$, $2 \leq i < m$, with $e_1$ or $e_2$).

Consider a minimum Manhattan network connecting the vertices in $V$. What do we know about the edges $E_{opt}$ of such a network?

1. $E_{opt}$ must include a Manhattan path between $v_1$ and $v$ within $R_1$. Note that this path has the same length as $e_1$.
2. $E_{opt}$ must include a Manhattan path between $v_m$ and $v$ within $[v, v_m]$. Note that within $R_2$ this path has at least the same weight as $e_2$.

3. If $m>2$ then there must be a network $N_{opt}$ connecting $v_i$, $2\leq i<m$, with $e_1$ or $e_2$ within $R_3$. Hence, it remains to prove that a minimum weight network connecting $v_i$, $2\leq i<m$, to the right or bottom side of $R_3$ has weight equal to a minimum weight rectangulation of $R_3$.

**Lemma 5** *A minimum-weight network connecting $v_i$, $2\leq i<m$, with $e_1$ or $e_2$ has length at least equal to the length of a minimum-weight rectangulation of the $C$-hull.*

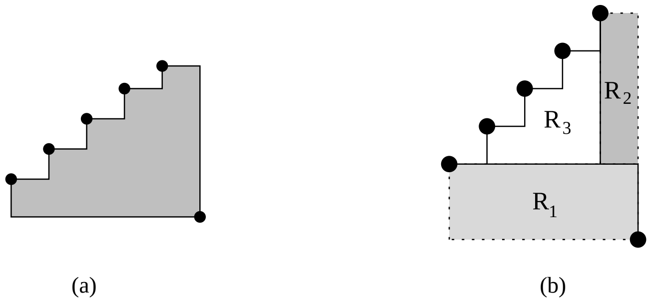



**Fig. 5.** (a) The charging area. (b) Partitioning the charging area into three regions.

Lemma 5 is a direct consequence of the following two lemmas. Consider a grid induced by the vertices of $V'$. Let $N_{opt}$ be an optimal network within $R_3$ connecting $v_i$, $2\leq i<m$, to the right or bottom side of $R_3$. Let $N_{rect}$ be a minimum-weight network, connecting $v_i$, $2\leq i<m$, to the right or bottom side of $R_3$, whose segments lie (only) on the grid induced by the vertices in $V'$.

**Lemma 6** $|N_{rect}|=|N_{opt}|$.

*Proof.* Let $P$ be a Manhattan path between vertices $v_i$ and $v$. Assume that the path is moving right on the grid and that it changes direction downwards without reaching an intersection in the grid. Denote by $t$ the last intersection of the grid that the path passed. Firstly, it is obvious that a path from $v_i$ to $v$ lying on the grid is of equal weight. Hence, the only reason to change direction "outside" the grid is that other paths may use this segment. Since all paths that may use this segment start at vertices to the left of $t$, they must start from vertices $v_j$, where $j<i$, which cannot intersect till the next horizontal grid line below it. Thus the path can be straightened out within that grid cell to follow the grid lines without decreasing its length. Going step by step, all paths can be modified to follow grid lines. Hence, the observation follows. □

**Lemma 7** *There exists a minimum-weight rectangulation of the interior of the $C$-hull of length $|N_{rect}|$.*

*Proof.* Every path between an interior point $v_i$ and $v$ moves (seen from $v_i$) only in two directions, right and down. The only case when a path would induce a non-rectangular network of the $C$-hull is when it turns right or down without meeting another path. Assume we follow a path from $v_i$ going down and then turning right, without meeting a horizontal segment. In this case the path could have been shortened by not changing direction, see Fig. 6(b), or by starting going right from the beginning. This is easy to see since the horizontal distance between a vertical segment and the turning point of the path is equal to the horizontal distance to the vertical segment and $v_i$. Hence, we do not gain anything by going downwards if the path is not meeting a horizontal segment. This means that there exists a rectangulation of the interior of the $C$-hull of weight $|N_{rect}|$. □

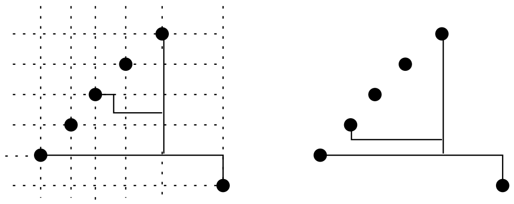


**Fig. 6.**

Putting these results together we obtain the following lemma.

**Lemma 1.** $(\sum_{v \in V} |E'_1(v)|) \leq r \times |E_{opt}|$.

To obtain the approximation factor for the algorithm just multiply the approximation factor for each step by four, since there are four sweeps. We summarize this paper by giving the main theorem.

**Theorem 8.** *Given a set of n points in the plane, and given an r-approximate, $R(n)$-time algorithm to compute a minimum-weight rectangulation of a staircase polygon, there exists an $O(n \log n + R(n))$-time algorithm that outputs a Manhattan network of total length at most $4r$ times that of a minimum Manhattan network.*

## 4 Conclusions and Open Problems

We introduce the concept of Manhattan networks on a given set of points; these networks guarantee Manhattan paths between every pair of points from the input set. We present efficient 4- and 8-approximation algorithms for the problem of computing minimum Manhattan networks. We conjecture that an efficient 2-approximate algorithm and a polynomial-time approximation scheme does exist for the problem. The following problems remain open:

1. Determine the complexity of the problem of computing minimum Manhattan networks.
2. Design a PTAS for the problem of computing minimum Manhattan networks.
3. Design a 2-approximate algorithm for the problem of computing minimum Manhattan networks.

## References


1. I. Althöfer, G. Das, D. P. Dobkin, D. Joseph, and J. Soares. On sparse spanners of weighted graphs. *Discrete Comput. Geom.*, 9:81–100, 1993.
2. Sanjeev Arora, Michelangelo Grigni, David Karger, Philip Klein, and Andrzej Woloszyn. A polynomial-time approximation scheme for weighted planar graph TSP. In *Proc. 9th ACM-SIAM Sympos. Discrete Algorithms*, pages 33–41, 1998.
3. S. Arya, G. Das, D. M. Mount, J. S. Salowe, and M. Smid. Euclidean spanners: short, thin, and lanky. In *Proc. 27th Annu. ACM Sympos. Theory Comput.*, pages 489–498, 1995.
4. B. Chandra, G. Das, G. Narasimhan, and J. Soares. New sparseness results on graph spanners. *Internat. J. Comput. Geom. Appl.*, 5:125–144, 1995.
5. G. Das and G. Narasimhan. A fast algorithm for constructing sparse Euclidean spanners. *Internat. J. Comput. Geom. Appl.*, 7:297–315, 1997.
6. C. Levcopoulos, G. Narasimhan, and M. Smid. Efficient algorithms for constructing fault-tolerant geometric spanners. In *Proc. 30th Annu. ACM Sympos. Theory Comput.*, pages 186–195, 1998.
7. C. Levcopoulos and A. Östlin. Linear-time heuristics for minimum weight rectangulation. In *5th Scandinavian Workshop on Algorithm Theory*, volume 1097 of *LNCS*, pages 271–283, 1996.
8. A. Lingas, R. Pinter, R. Rivest, and A. Shamir. Minimum edge length partitioning of rectilinear polygons. In *Proc. 20th Allerton Conf. Commun. Control Comput.*, pages 53–63, 1982.
9. S. B. Rao and W. D. Smith. Approximating geometrical graphs via "spanners" and "banyans". In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, 1998.