The Minimum Manhattan Network Problem— Approximations and Exact Solutions

Marc Benkert [a], Takeshi Shirabe [b], and Alexander Wolff [a]

[a] *Faculty of Computer Science, Karlsruhe University, Germany. WWW: i11www.ilkd.uka.de/algo/group*

[b] *Institute for Geoinformation, Technical University of Vienna, Austria. Email: shirabe@geoinfo.tuwien.ac.at*

1. Introduction

A *Manhattan p–q path* is a geodesic in the Manhattan (or L_1-) metric that connects p and q, i.e. a staircase path between p and q. Given a set of points P in the plane, a *Manhattan network* is a set of axis-parallel line segments that contains a Manhattan p–q path for each pair $\{p, q\}$ of points in P.

In this paper we consider the *minimum* Manhattan network problem which consists of finding a Manhattan network of minimum total length, an *MMN* in short, i.e. a 1-spanner for the Manhattan metric. The problem is likely to have applications in VLSI layout. Its complexity status is unknown.

The problem has been considered before. Gudmundsson et al. [1] have proposed a factor-8 $O(n \log n)$-time and a factor-4 $O(n^3)$-time approximation algorithm, where n is the number of input points. Later Kato et al. [2] have given a factor-2 $O(n^3)$-time approximation algorithm. However, their correctness proof is incomplete.

In this paper we give a geometric factor-3 approximation algorithm that runs in $O(n \log n)$ time and the first mixed-integer programming (MIP) formulation for the MMN problem. We have implemented and evaluated both approaches.

2. Preliminaries

We will use the notion of a *generating set* that has been introduced in [2]. A generating set is a subset Z of $\binom{P}{2}$ with the property that a network containing Manhattan paths for all pairs in Z is a Manhattan network of P.

The authors of [2] defined a generating set Z with the nice property that Z consists only of a linear number of point pairs. Here we use the same generating set Z, but more intuitive names for the subsets of Z. In [2] $Z = Z_h \cup Z_v \cup Z_x \cup Z_y \cup Z_2$. Here we define $Z = Z_{\mathrm{hor}} \cup Z_{\mathrm{ver}} \cup Z_{\mathrm{quad}}$, where $Z_{\mathrm{hor}} = Z_h \cup Z_y$, $Z_{\mathrm{ver}} = Z_v \cup Z_x$, and $Z_{\mathrm{quad}} = Z_2$. We consider Z_{quad} a set of ordered pairs.

Now let $\mathcal{R}_{\mathrm{hor}} = \{\mathrm{BBox}(p, q) \mid \{p, q\} \in Z_{\mathrm{hor}}\}$, where $\mathrm{BBox}(p, q)$ is the smallest axis-parallel closed rectangle that contains p and q. Note that $\mathrm{BBox}(p, q)$ is just the line segment $\overline{pq}$ if p and q lie on the same horizontal or vertical line. In this case we consider $\mathrm{BBox}(p, q)$ a *degenerate rectangle.* Define $\mathcal{R}_{\mathrm{ver}}$ and $\mathcal{R}_{\mathrm{quad}}$ analogously. Let $\mathcal{A}_{\mathrm{hor}}$, $\mathcal{A}_{\mathrm{ver}}$, and $\mathcal{A}_{\mathrm{quad}}$ be the subsets of the plane that are defined by the union of the rectangles in $\mathcal{R}_{\mathrm{hor}}$, $\mathcal{R}_{\mathrm{ver}}$, and $\mathcal{R}_{\mathrm{quad}}$, respectively. As Kato et al. we start with some basic, yet incomplete network whose length is bounded by the length of an MMN.

Definition 1 *[2] A set of vertical line segments $\mathcal{V}$* covers *$\mathcal{R}_{\mathrm{ver}}$, if for any horizontal line ℓ and any $R \in \mathcal{R}_{\mathrm{ver}}$ with $R \cap \ell \neq \emptyset$ there is a $V \in \mathcal{V}$ with $V \cap \ell \neq \emptyset$. We say that $\mathcal{V}$ is a* minimum vertical cover (MVC) *if $\mathcal{V}$ has minimum length among all covers of $\mathcal{R}_{\mathrm{ver}}$. The definition of a* minimum horizontal cover (MHC) *is analogous.*

Kato et al. have observed the following.

Lemma 2 *[2] The union of an* MVC *and an* MHC *has length bounded by the length of an* MMN.

In general such a union does not satisfy, i.e. connect by Manhattan paths, all pairs in Z_{ver} and Z_{hor}. Additional segments must be added to the union to achieve this. To ensure that the total length of these segments can be bounded, we need covers with a special property. Obviously the segments in an MVC can be moved such that each segment is contained in a vertical edge of a rectangle in $\mathcal{R}_{\mathrm{ver}}$. We say that an MVC is *nice* if additionally each cover segment is incident to a point in P. Note that each vertical rectangle edge contains at most one segment of a nice MVC, since degenerate rectangles do not share edges with other rectangles and must therefore be covered completely. In order to show that every point set has in fact a nice MVC, we need the following definitions.

For a horizontal line ℓ consider the graph $G_\ell(V_\ell, E_\ell)$, where V_ℓ is the intersection of ℓ with the vertical edges of rectangles in $\mathcal{R}_{\text{ver}}$, and there is an edge in E_ℓ if two intersection points belong to the same rectangle. We say that a point v in V_ℓ is *odd* if the number of points to the left of v that belong to the same connected component of G_ℓ is odd, otherwise v is even. For a vertical edge e of a rectangle in $\mathcal{R}_{\text{ver}}$, let an *odd segment* be an inclusion-maximal connected set of odd points on e. Define *even segments* accordingly. For example, the segment s (drawn bold in Figure 1) of the edge f is an even segment, while $f \setminus s$ is odd. We say that the parity of an edge changes where two segments of different parity touch.

Theorem 3 *Every point set P has a nice* MVC *and a nice* MHC*.*

PROOF. We only show the statement for the vertical case, the horizontal case is analogous. Our proof is constructive. Let $\mathcal{V}$ be the union of all odd segments and all degenerate rectangles in $\mathcal{R}_{\text{ver}}$. Clearly $\mathcal{V}$ covers $\mathcal{R}_{\text{ver}}$. Let ℓ be a horizontal that intersects $\mathcal{A}_{\text{ver}}$. Consider a connected component C of G_ℓ and let k be the number of vertices in C. If k is even then any cover must contain at least $k/2$ vertices of C, and $\mathcal{V}$ contains exactly $k/2$. On the other hand, if $k > 1$ is odd then any cover must contain at least $(k-1)/2$ vertices of C, and $\mathcal{V}$ contains exactly $(k-1)/2$. Thus $\mathcal{V}$ is an MVC.

To see that $\mathcal{V}$ is nice, we consider a vertical edge e of a rectangle in $\mathcal{R}_{\text{ver}}$ and the input point p_0 on e. We show that either e is even or p_0 lies on the only odd segment of e. In both cases $\mathcal{V}$ contains only cover segments that touch an input point.

Wlog. let p_0 be the topmost point of e. Let $p_0, p_1, \ldots, p_k$ be the input points in order of decreasing x-coordinate that span the rectangles in $\mathcal{R}_{\text{ver}}$ that are relevant for the parity of e. Let $p_i = (x_i, y_i)$ and $\overline{y_0} = y_0$, $\overline{y_1} = y_1$. For $2 \le i \le k$ define recursively $\overline{y_i} = \min\{y_i, \overline{y_{i-2}}\}$ if i is even, and $\overline{y_i} = \max\{y_i, \overline{y_{i-2}}\}$ if i is odd. Let $\overline{p_i} = (x_i, \overline{y_i})$, and let $\overline{\mathcal{L}}$ be the polygonal chain through $p_0, p_1, \overline{p_2}, \overline{p_3}, \ldots, \overline{p_k})$ in this order, see Figure 1. Note that the parity of a point v on e is determined by the number of segments of $\overline{\mathcal{L}}$ that intersect the horizontal h through v.

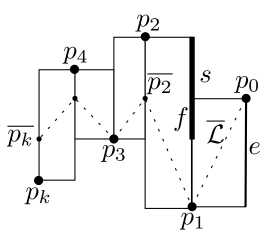


Fig. 1. Proof of Theorem 3.

If h is below $\overline{p_k}$, then it intersects a descending segment for each ascending segment of $\overline{\mathcal{L}}$, hence v is even. If on the other hand h goes through or is above $\overline{p_k}$, then it intersects an ascending segment for each descending segment—plus $\overline{p_1 p_0}$, hence v is odd. So e can change parity only in $(x_0, \overline{y_k})$. $\square$

A simple sweep-line algorithm yields the following.

Lemma 4 *A nice* MVC *and a nice* MHC *can be computed in $O(n \log n)$ time using linear space.*

3. An Approximation Algorithm

Our algorithm ApproxMMN proceeds in three phases, see Algorithm 1. In phase I we compute the generating set $Z_{\text{ver}} \cup Z_{\text{hor}} \cup Z_{\text{quad}}$. In phase II we satisfy all pairs in $Z_{\text{ver}} \cup Z_{\text{hor}}$ by computing a nice MVC $\mathcal{C}_{\text{ver}}$ and a nice MHC $\mathcal{C}_{\text{hor}}$, and by then adding at most one additional line segment for each rectangle $\mathcal{R}_{\text{ver}} \cup \mathcal{R}_{\text{hor}}$. Since each rectangle $R = \text{BBox}(p, q) \in \mathcal{R}_{\text{ver}}$ ($\mathcal{R}_{\text{hor}}$) is covered nicely, it suffices to add a horizontal (vertical) segment whose length is the width (height) of R in order satisfy $\{p, q\}$. Let $\mathcal{S}$ be the set of these additional segments. Consider the vertical strip that is defined by a rectangle $R \in \mathcal{R}_{\text{ver}}$. By definition of $\mathcal{R}_{\text{ver}}$, R is the only rectangle in $\mathcal{R}_{\text{ver}}$ that intersects the interior of the strip. Thus the total length of the additional horizontal (vertical) segments is the width W (height H) of $\text{BBox}(P)$. By Lemma 2 the network $N_1 = \mathcal{C}_{\text{ver}} \cup \mathcal{C}_{\text{hor}} \cup \mathcal{S}$ has length $\le |N_{\text{opt}}| + H + W$, where N_{opt} is a fixed MMN and $|M|$ is the total length of a set M of line segments.

In phase III we satisfy the pairs in Z_{quad}. Let $Q(r, 1) = \{s \in \mathbb{R}^2 \mid x_r < x_s \text{ and } y_r < y_s\}$ be the first quadrant of the Cartesian coordinate system with origin r. Define $Q(r, 2)$, $Q(r, 3)$, $Q(r, 4)$ analogously and in the usual order. Let $P(q, t) = \{p \in P \cap Q(q, t) \mid (p, q) \in Z_{\text{quad}}\}$ for $t = 1, 2, 3, 4$. Let $\Delta(q, t) = \bigcup_{p \in P(q,t)} \text{BBox}(p, q) \setminus \text{int}(\mathcal{A}_{\text{hor}} \cup \mathcal{A}_{\text{ver}})$, where $\text{int}(M)$ denotes the interior of a set $M \subseteq \mathbb{R}^2$. Let $\delta(q, t)$ be the union of those connected components of $\Delta(q, t)$ that are incident to some $p \in P(q, t)$. Note that each connected component A of $\delta(q, t)$ is a staircase polygon—by definition of Z_{quad} there is a strictly x- and y-monotone ordering of the points in $P(q, t)$. The C-hull of A is the union of A and the bounding boxes of neighboring

Algorithm 1 ApproxMMN

Phase I: Compute $Z = Z_{\text{ver}} \cup Z_{\text{hor}} \cup Z_{\text{quad}}$.
Phase II: Satisfy $Z_{\text{ver}} \cup Z_{\text{hor}}$:
  compute a nice MVC $\mathcal{C}_{\text{ver}}$ and a nice MHC $\mathcal{C}_{\text{hor}}$
  compute set $\mathcal{S}$ of additional horizontal (vertical) segments for rectangles in $\mathcal{R}_{\text{ver}}$ ($\mathcal{R}_{\text{hor}}$)
  $N_1 \leftarrow \mathcal{C}_{\text{ver}} \cup \mathcal{C}_{\text{hor}} \cup \mathcal{S}$, $N_2 \leftarrow \emptyset$, $N_3 \leftarrow \emptyset$
Phase III: Satisfy Z_{quad}:
  for each region δ of type $\delta(q,t)$ **do**
    for each connected component A of δ **do**
      compute rectangulation $R_{A'}$ of A'
      $N_2 \leftarrow N_2 \cup \partial A' \cup \{s_{A'}\}$
      $N_3 \leftarrow N_3 \cup (R_{A'} \setminus \partial A')$
    end for
  end for
return $N = N_1 \cup N_2 \cup N_3$

input points on the boundary ∂A of A. It is known that any MMN rectangulates the C-hull of A [1, Lemma 4]. The same holds for a slightly smaller region A' that can be connected to N_1 via at most one segment $s_{A'}$. There is a simple $O(n)$-time factor-2 approximation algorithm B for rectangulating staircase polygons [1]. We use B to compute the rectangulation $R_{A'}$ of each A'. Let N_2 be the union of all $\partial A'$ and all $s_{A'}$. Let N_3 be the union of the rectangulations $R_{A'}$ without $\partial A'$. Our algorithm returns the line segments in $N = N_1 \cup N_2 \cup N_3$.

To bound the length of N we partition the plane into two regions and compare N to N_{opt} in each region separately. Region $\mathcal{A}_3$ is the union of $\text{int}(A')$ over all areas of type A', while $\mathcal{A}_{12} = \mathbb{R}^2 \setminus \mathcal{A}_3$. We have $N_1 \cup N_2 \subseteq \mathcal{A}_{12}$ and $N_3 \subseteq \mathcal{A}_3$, and the interiors of different regions of type A do not intersect. On the one hand the approximation factor of B yields that $|N \cap \mathcal{A}_3| \leq 2|N_{\text{opt}} \cap \mathcal{A}_3|$. On the other hand we can show that $|N_2| \leq 2|N_{\text{opt}}| - (H + W)$. Thus $|N \cap \mathcal{A}_{12}| = |(N_1 \cup N_2) \cap \mathcal{A}_{12}| \leq 3|N_{\text{opt}} \cap \mathcal{A}_{12}|$, which in turn yields $|N| \leq 3|N_{\text{opt}}|$.

Theorem 5 *A 3-approximation of an* MMN *can be computed in $O(n \log n)$ time and $O(n)$ space.*

4. A MIP Formulation

In this section we give the first MIP formulation of the MMN problem. This formulation gives us the possibility to implement an exact solver for the MMN problem that can solve small examples in a bearable amount of time. Those will be used as benchmarks for our approximation algorithm.

We need some notation: For a set P of n input points $p_1(x_1, y_1), \ldots, p_n(x_n, y_n)$ let $x^1 < \cdots < x^u$ and $y^1 < \cdots < y^w$ be the ascending sequences of x- respectively y-coordinates of the input points. The grid Γ induced by P consists of the *grid points* (x^i, y^j) with $i = 1, \ldots, u$ and $j = 1, \ldots, w$. In this section we will only consider pairs $\{p, q\} \in Z$ with $x_p \leq x_q$. This is no restriction since we can flip the names of p and q. For each such pair let $V(p,q) = \Gamma \cap \text{BBox}(p,q)$ and let $A(p,q)$ be the set of arcs between horizontally or vertically adjacent grid points in $V(p,q)$. Horizontal arcs are always directed from left to right, vertical arcs point upwards (downwards) if $y_p < y_q$ ($y_p > y_q$). Our formulation is based on the *grid graph* $G_P(V, A)$, where $V = \bigcup_{\{p,q\} \in Z} V(p,q)$ and $A = \bigcup_{\{p,q\} \in Z} A(p,q)$. Let $E = \{\{g, g'\} \mid (g, g') \in A \text{ or } (g', g) \in A\}$ be the set of undirected edges.

For each pair $\{p, q\} \in Z$ we enforce the existence of a p–q Manhattan path by a flow model as follows. We introduce one 0–1 variable $f(p, q, g, g')$ for each arc (g, g') in $A(p,q)$, which encodes the size of the flow along arc (g, g'). For each grid point g in $V(p,q)$ we introduce the flow constraint

$$\left.\begin{array}{l} \displaystyle\sum_{(g,g') \in A(p,q)} f(p,q,g,g') \\ \displaystyle - \sum_{(g',g) \in A(p,q)} f(p,q,g',g) \end{array}\right\} = \begin{cases} +1 & \text{if } g = p, \\ -1 & \text{if } g = q, \\ 0 & \text{else.} \end{cases} \quad (1)$$

Next we introduce a continuous variable $F(g, g')$ for each edge $\{g, g'\}$ in E. This variable will in fact be forced to take a 0–1 value by the objective function and the following constraints. The MMN that we want to compute will consist of all grid edges $\{g, g'\}$ with $F(g, g') = 1$. We now add one or two constraints for each $\{g, g'\}$ in E and each $\{p, q\} \in Z$ with $\overline{gg'} \subseteq \text{BBox}(p,q)$:

$$F(g, g') \quad \geq \quad \begin{cases} f(p,q,g,g') & \text{if } (g,g') \in A, \\ f(p,q,g',g) & \text{if } (g',g) \in A. \end{cases} \quad (2)$$

Note that the two conditions are not mutually exclusive. Our objective function expresses the total length of the selected grid edges:

$$\min! \sum_{\{g,g'\} \in E} |gg'| \cdot F(g, g'), \quad (3)$$

where $|gg'|$ is the Euclidean distance of g and g'.

This MIP formulation uses $O(n^3)$ variables and constraints. By treating pairs in Z_{quad} more carefully, a reduction to $O(n^2)$ is possible, see full paper. It is not hard to see that our formulation always yields an MMN:

Theorem 6 *Let P be a set of points and let Z, A, and E be defined as above. Let $F : E \to \mathbb{R}_0^+$ and $f : Z \times A \to \{0,1\}$ be functions that fulfill (1) & (2) and minimize (3). Then the set of line segments $\{\overline{gg'} \mid \{g,g'\} \in E,\ F(g,g') \geq 1\}$ is an* MMN *of P.*

Due to our objective function (3), Equation (1) can be replaced by an inequality (with direction $\geq$). If the resulting constraint matrix was totally unimodular (every square submatrix has determinant in $\{-1,0,+1\}$), every vertex of the solution polyhedron would be integral and the MMN problem would in fact correspond to an LP. Unfortunately it turned out that this is not the case and that there are instances with fractional vertices that minimize our objective function.

5. Experiments

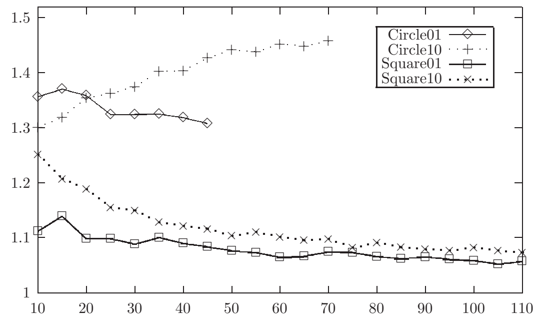
We used two different types of random instances.

SQUAREk instances were generated by drawing n different points with uniform distribution from a $kn \times kn$ integer grid. We wanted to see the effects of having more (k small) or less (k large) points with the same x- or y-coordinate. If a pair of points shares a coordinate, the manhattan path connecting them is uniquely determined.

CIRCLEk instances consist of a point p_1 at the origin and $n-1$ points on the upper half of the unit circle. The points are distributed as follows. The interval $I = [0, \pi/4]$ is split into k subintervals $I_1, \ldots, I_k$ of equal length. We used $k \in \{1,2,5,10\}$. Then $n-1$ random numbers $r_2, \ldots, r_n$ are drawn from I. If the number r_i falls into a subinterval of even index, it is mapped to the point $p_i = (\cos r_i, \sin r_i)$ otherwise to $p_i = (-\cos r_i, \sin r_i)$. The resulting points p_i (except for the topmost point in each quadrant and the "bottommost" point in each subinterval) all form pairs $\{p_i, p_1\}$ that are in Z_{quad}. This makes CIRCLE instances very different from SQUARE instances where only few point pairs belong to Z_{quad}.

We generated instances of the above types and solved them with APPROXMMN and with Cplex using the MIP formulation of Section 4. We implemented APPROXMMN in C++ using the compiler gcc-3.3. The asymptotic runtime of our implementation is $\Theta(n^2)$, the real runtime was measured on an AMD Athlon 1800+ with 512 MB RAM under Linux-2.4.20. To compute exact solutions we used the LP Barrier Solver of ILOG Cplex-9.0 on an IBM RS/6000. The results of our experiments can be found in the diagram below. The sample size, i.e. the number of points per instance, is shown on the x-axis. For each sample size we generated 30 instances and averaged the results over those. The y-axis shows the performance ratio of APPROXMMN, i.e. the ratio of the length of the network computed by APPROXMMN over the length of the MMN computed by Cplex.

Cplex ran out of memory on CIRCLE01 instances of more than 45 points and on SQUARE10 instances of more than 110 points. Below these thresholds APPROXMMN always had a performance ratio below 1.55, which is much better than what the approximation factor of 3 suggests. While the ratio seems to approach 1 on SQUARE instances of increasing size, the picture is not so clear for CIRCLE instances:



The runtime of APPROXMMN was practically independent of the type of instance. The CPU times we measured reflected the quadratic asymptotic runtime. 500 points took roughly 0.3 seconds.

The exact solver depended much more on the type of instance than the approximation algorithm. It solved SQUARE instances much faster than CIRCLE instances. This is due to the fact that pairs in Z_{quad} require a quadratic number of variables and constraints in our MIP formulation, while pairs in Z_{ver} and Z_{hor} need only a linear number. 100 points of type SQUAREk took 0.6–1.6 seconds and 6–13 seconds for $k = 1$ and 10, respectively, while 40 points of type CIRCLEk took 61–480 seconds and 1.2–2.4 seconds for $k = 1$ and 10, respectively.

References

[1] J. Gudmundsson, C. Levcopoulos, and G. Narasimhan. Approximating a minimum Manhattan network. *Nordic J. Comput.*, 8:219–232, 2001.

[2] R. Kato, K. Imai, and T. Asano. An improved algorithm for the minimum Manhattan network problem. In *Proc. ISAAC'02*, vol. 2518 of *LNCS*, pages 344–356, 2002.