The Minimum Manhattan Network Problem: A Fast Factor-3 Approximation*

Marc Benkert, Alexander Wolff, and Florian Widmann

Faculty of Computer Science, Karlsruhe University, P.O. Box 6980,
D-76128 Karlsruhe, Germany
i11www.ira.uka.de/algo/group

Abstract. Given a set of nodes in the plane and a constant $t \geq 1$, a Euclidean *t-spanner* is a network in which, for any pair of nodes, the ratio of the network distance and the Euclidean distance of the two nodes is at most t. These networks have applications in transportation or communication network design and have been studied extensively.

In this paper we study 1-spanners under the Manhattan (or L_1-) metric. Such networks are called *Manhattan networks*. A Manhattan network for a set of nodes can be seen as a set of axis-parallel line segments whose union contains an x- and y-monotone path for each pair of nodes. It is not known whether it is NP-hard to compute minimum Manhattan networks, i.e. Manhattan networks of minimum total length. In this paper we present a factor-3 approximation algorithm for this problem. Given a set of n nodes, our algorithm takes $O(n \log n)$ time and linear space.

1 Introduction

For many applications it is desirable to connect the nodes of a transportation or communication network by short paths within the network. In the Euclidean plane this can be achieved by connecting all pairs of nodes by straight line segments. While the complete graph minimizes node-to-node travel time, it maximizes the network-construction costs. An interesting alternative are Euclidean *t-spanners*, i.e. networks in which the ratio of the network distance and the Euclidean distance between any pair of nodes is bounded by a constant $t \geq 1$. Euclidean spanners have been studied extensively, and researchers have tried to combine the spanner property with other desirable properties, such as small node degree, small total edge length, and small diameter. Euclidean spanners with one or more of these properties can be constructed in $O(n \log n)$ time [1], where n is the number of nodes.

Under the Euclidean metric, in a 1-spanner (which is the complete graph) the location of each edge is uniquely determined. This is not the case in the Manhattan (or L_1-) metric, where an edge $\{p, q\}$ of a 1-spanner is a *Manhattan p–q path*, i.e. an x- and y-monotone path between p and q. A 1-spanner under the Manhattan metric for a set $P \subset \mathbb{R}^2$ is called a *Manhattan network* (MMN)

* This work was supported by grant WO 758/4-1 of the German Science Foundation.

and can be seen as a set of axis-parallel line segments whose union contains a Manhattan p–q path for each pair $\{p, q\}$ of points in P.

In this paper we investigate how the extra degree of freedom in routing edges can be used to construct Manhattan networks of minimum total length, so-called *minimum Manhattan networks* (MMN). The MMN problem may have applications in city planning or VLSI layout. Lam et al. [5] also describe an interesting application in computational biology. Approximation algorithms for the MMN problem have been considered before. Gudmundsson et al. [3] have proposed an $O(n \log n)$-time factor-8 and an $O(n^3)$-time factor-4 approximation algorithm. Later, Kato et al. [4] have given an $O(n^3)$-time factor-2 approximation algorithm. However, their correctness proof is incomplete. It is not known whether the MMN problem is NP-hard.

In this paper we present an $O(n \log n)$-time factor-3 approximation algorithm. We use some of the ideas of [4], but our algorithm is simpler, faster and uses only linear (instead of quadratic) storage. The main novelty of our approach is that we partition the plane into two regions and compare the network computed by our algorithm to an MMN in each region separately.

This paper is structured as follows. In Section 2 we give some basic definitions and observations. In Section 3 we show how the backbone of our network is computed. We describe the algorithm in Section 4 and analyze it in Section 5.

2 Preliminaries

We use $|M|$ to denote the total length of a set M of line segments. For all such sets M we assume throughout the paper that each segment of M is inclusion-maximal with respect to $\bigcup M$. It is not hard to see that for every Manhattan network M there is a Manhattan network M' with $|M'| \leq |M|$ that is contained in the grid induced by the input points, i.e. M' is a subset of the union U of the horizontal and vertical lines through the input points. Therefore we will only consider networks contained in U. It is clear that any meaningful Manhattan network of a point set P is contained in the bounding box $\mathrm{BBox}(P)$ of P. Finding a Manhattan network for given P is trivial, e.g. the parts of U within $\mathrm{BBox}(P)$ yield a Manhattan network. However, this network can be n times longer than an MMN, as the point set $\{(1, 1), \dots, (n, n)\}$ shows.

We will use the notion of a *generating set* [4]. A generating set Z is a subset of $P \times P$ with the property that a network containing Manhattan paths for all pairs in Z is already a Manhattan network of P. The authors of [4] define a linear-size generating set Z. We use the same generating set Z, but more intuitive names for the subsets of Z. We define $Z = Z_{\mathrm{hor}} \cup Z_{\mathrm{ver}} \cup Z_{\mathrm{quad}}$. These subsets are defined below. Our algorithm will establish Manhattan paths for all point pairs of Z—first for those in $Z_{\mathrm{hor}} \cup Z_{\mathrm{ver}}$ and then for those in Z_{quad}.

Definition 1 (Z_{ver}). *Let $P = \{p_1, \dots, p_n\}$ be the set of input points in lexicographical order, where $p_i = (x_i, y_i)$. Let $x^1 < \cdots < x^u$ be the sequence of x-coordinates of the points in P in ascending order. For $i = 1, \dots, u$ let $P^i = \{p_{a(i)}, p_{a(i)+1}, \dots, p_{b(i)}\}$ be the set of all $p \in P$ with x-coordinate x^i. Then*

$$\begin{aligned} Z_{\text{ver}} = \; & \{(p_i, p_{i+1}) \mid x_i = x_{i+1} \textit{ and } 1 \le i < n\} \\ & \cup \{(p_{a(i)}, p_{b(i+1)}) \mid y_{a(i)} > y_{b(i+1)} \textit{ and } 1 \le i < u\} \\ & \cup \{(p_{b(i)}, p_{a(i+1)}) \mid y_{b(i)} < y_{a(i+1)} \textit{ and } 1 \le i < u\}. \end{aligned}$$

In Figure 1 all pairs of Z_{ver} are connected by an edge. Note that Z_{ver} consists of at most $n-1$ point pairs. If no points have the same x-coordinate, then $Z_{\text{ver}} = \{(p_i, p_{i+1}) \mid 1 \le i < n\}$, i.e. Z_{ver} is the set of neighboring pairs in the lexicographical order. The definition of Z_{hor} is analogous to that of Z_{ver} with the roles of x and y exchanged. Figure 2 shows that $Z_{\text{hor}} \cup Z_{\text{ver}}$ is not always a generating set: Since $(p, h) \in Z_{\text{hor}}$ and $(p, v) \in Z_{\text{ver}}$, no network that consists only of Manhattan paths between pairs in $Z_{\text{hor}} \cup Z_{\text{ver}}$ contains a Manhattan p–q path. This shows the necessity of a third subset Z_{quad} of Z.

Definition 2 (Z_{quad}). *For a point $r \in \mathbb{R}^2$ denote its Cartesian coordinates by (x_r, y_r). Let $Q(r,1) = \{s \in \mathbb{R}^2 \mid x_r \le x_s \text{ and } y_r \le y_s\}$ 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. Then Z_{quad} is the set of all ordered pairs $(p,q) \in P \times P$ with $q \in Q(p,t) \setminus \{p\}$ and $t \in \{1,2,3,4\}$ that fulfill*

(a) q is the point that has minimum y-distance from p among all points in $Q(p,t) \cap P$ with minimum x-distance from p, and
(b) there is no $q' \in Q(p,t) \cap P$ with (p,q') or (q',p) in $Z_{\text{hor}} \cup Z_{\text{ver}}$.

Obviously Z_{quad} consists of at most $4n$ point pairs.

For our analysis we need the following regions of the plane. Let $\mathcal{R}_{\text{hor}} = \{\text{BBox}(p,q) \mid \{p,q\} \in Z_{\text{hor}}\}$, where $\text{BBox}(p,q)$ is the smallest axis-parallel closed rectangle that contains p and q. Note that $\text{BBox}(p,q)$ is just the line segment $\text{Seg}[p,q]$ from p to q, if p and q lie on the same horizontal or vertical line. In this case we call $\text{BBox}(p,q)$ a *degenerate rectangle.* Define $\mathcal{R}_{\text{ver}}$ and $\mathcal{R}_{\text{quad}}$ analogously. Let $\mathcal{A}_{\text{hor}}$, $\mathcal{A}_{\text{ver}}$, and $\mathcal{A}_{\text{quad}}$ be the subsets of the plane that are defined by the union of the rectangles in $\mathcal{R}_{\text{hor}}$, $\mathcal{R}_{\text{ver}}$, and $\mathcal{R}_{\text{quad}}$, respectively.

Any Manhattan network has to bridge the vertical (horizontal) gap between the points of each pair in Z_{ver} (Z_{hor}). Of course this can be done such that at the same time the gaps of adjacent pairs are (partly) bridged. The corresponding minimization problem is defined as follows.

Definition 3 (Kato et al. [4]). *A set of vertical line segments $\mathcal{V}$ is a* cover *of (or* covers*) $\mathcal{R}_{\text{ver}}$, if any $R \in \mathcal{R}_{\text{ver}}$ is covered, i.e. for any horizontal line ℓ with $R \cap \ell \neq \emptyset$ there is a $V \in \mathcal{V}$ with $V \cap \ell \cap R \neq \emptyset$. We say that $\mathcal{V}$ is a* minimum vertical cover (MVC) *if $\mathcal{V}$ has minimum length among all covers of $\mathcal{R}_{\text{ver}}$. The definition of a* minimum horizontal cover (MHC) *is analogous.*

For an example, see Figure 3. Since any MMN covers $\mathcal{R}_{\text{ver}}$ and $\mathcal{R}_{\text{hor}}$, we have:

Lemma 1 (Kato et al. [4]). *The union of an* MVC *and an* MHC *has length bounded by the length of an* MMN.

To sketch our algorithm we need the following notation. Let N be a set of line segments. We say that N *satisfies* a set of point pairs S if N contains a Manhattan p–q path for each $\{p, q\} \in S$. We use $\bigcup N$ to denote the corresponding set of points, i.e. the union of the line segments in N. Let ∂M be the boundary of a set $M \subseteq \mathbb{R}^2$.

Our algorithm proceeds in four phases. In phase 0, we compute Z. In phase I, we construct a network N_1 that contains the union of a special MVC and a special MHC and satisfies $Z_{\text{ver}} \cup Z_{\text{hor}}$. In phase II, we identify a set $\mathcal{R}$ of open regions in $\mathcal{A}_{\text{quad}}$ that do not intersect N_1, but need to be bridged in order to satisfy Z_{quad}. The regions in $\mathcal{R}$ are staircase polygons. They give rise to two sets of segments, N_2 and N_3, which are needed to satisfy Z_{quad}. For each region $A \in \mathcal{R}$ we put the segments that form $\partial A \setminus \bigcup N_1$ into N_2, plus, if necessary, an extra segment to connect ∂A to N_1. Finally, in phase III, we bridge the regions in $\mathcal{R}$ by computing a set N_3 of segments in the interior of the regions. This yields a Manhattan network $N = N_1 \cup N_2 \cup N_3$.

The novelty of our analysis is that we partition the plane into two areas and compare N to an MMN in each area separately. The area $\mathcal{A}_3$ consists of the interiors of the regions $A \in \mathcal{R}$ and contains N_3. The other area $\mathcal{A}_{12}$ is the complement of $\mathcal{A}_3$ and contains $N_1 \cup N_2$. For a fixed MMN N_{opt} we show that $|N \cap \mathcal{A}_{12}| \leq 3|N_{\text{opt}} \cap \mathcal{A}_{12}|$ and $|N \cap \mathcal{A}_3| \leq 2|N_{\text{opt}} \cap \mathcal{A}_3|$, and thus $|N| \leq 3|N_{\text{opt}}|$. The details will be given in Section 4.

We now define vertical and horizontal neighbors of points in P. Knowing these neighbors helps to compute Z and $\mathcal{R}$.

Definition 4 (Neighbors). *For a point $p \in P$ and $t \in \{1, 2, 3, 4\}$ let p.*xnbor$[t]$ $=$ nil *if $Q(p, t) \cap P = \{p\}$. Otherwise p.*xnbor$[t]$ *points to the point that has minimum y-distance from p among all points in $Q(p, t) \cap P \setminus \{p\}$ with minimum x-distance from p. The pointer p.*ynbor$[t]$ *is defined by exchanging x and y in the above definition.*

All pointers of types xnbor and ynbor can be computed by a simple plane sweep in $O(n \log n)$ time. Then we compute Z_{ver} by going through the points in lexicographical order and examining the pointers of type xnbor. This works analogously for Z_{hor}. Note that by Definition 1 each point $q \in P$ is incident to at most three rectangles of $\mathcal{R}_{\text{ver}}$, at most two of which can be (non-) degenerate. We refer to points $p \in P$ with $(p, q) \in Z_{\text{ver}}$ as *vertical predecessors* of q and to points $r \in P$ with $(q, r) \in Z_{\text{ver}}$ as *vertical successors* of q. We call a predecessor or successor of q *degenerate* if it has the same x-coordinate as q. Note that each point can have at most one degenerate vertical predecessor and successor, and at most one non-degenerate vertical predecessor and successor. Horizontal predecessors and successors are defined analogously with respect to Z_{hor}. For each $t \in \{1, 2, 3, 4\}$ the pair $(q, q.\text{xnbor}[t])$ lies in Z_{quad} if and only if $q.\text{xnbor}[t] \neq$ nil and no vertical or horizontal predecessor or successor of q lies in $Q(q, t)$. Thus:

Lemma 2. *All pointers of type* xnbor, ynbor *and the generating set Z can be computed in $O(n \log n)$ time.*

3 Minimum Covers

In general the union of an MVC and an MHC does not satisfy $Z_{\text{ver}} \cup Z_{\text{hor}}$. Additional segments must be added to achieve this. To ensure that the total length of these segments can be bounded, we need covers with a special property. We say that a cover is *nice* if each cover segment contains an input point.

Lemma 3. *For any nice* MVC $\mathcal{V}$ *and any nice* MHC $\mathcal{H}$ *there is a set* $\mathcal{S}$ *of line segments such that* $\mathcal{V} \cup \mathcal{H} \cup \mathcal{S}$ *satisfies* $Z_{\text{ver}} \cup Z_{\text{hor}}$ *and* $|\mathcal{S}| \leq W + H$*, where* W *and* H *denote width and height of* $\text{BBox}(P)$*, respectively. We can compute the set* $\mathcal{S}$ *in linear time if for each* $R \in \mathcal{R}_{\text{ver}}$ *(*$\mathcal{R}_{\text{hor}}$*) we have constant-time access to the segments in* $\mathcal{V}$ *(*$\mathcal{H}$*) that intersect* R*.*

Proof. We show that there is a set $\mathcal{S}_{\mathcal{V}}$ of horizontal segments with $|\mathcal{S}_{\mathcal{V}}| \leq W$ such that $\mathcal{V} \cup \mathcal{S}_{\mathcal{V}}$ satisfies Z_{ver}. Analogously it can be shown that there is a set $\mathcal{S}_{\mathcal{H}}$ of vertical segments with $|\mathcal{S}_{\mathcal{H}}| \leq H$ such that $\mathcal{H} \cup \mathcal{S}_{\mathcal{H}}$ satisfies Z_{hor}. This proves the lemma.

Let $(p, q) \in Z_{\text{ver}}$. If $R = \text{BBox}(p, q)$ is degenerate, then by the definition of a cover, there is a line segment $s \in \mathcal{V}$ with $R \subseteq s$, and thus $\mathcal{V}$ satisfies (p, q).

Otherwise R defines a non-empty vertical open strip $\sigma(p, q)$ bounded by p and q. Note that by the definition of Z_{ver}, R is the only rectangle in $\mathcal{R}_{\text{ver}}$ that intersects $\sigma(p, q)$. This yields that the widths of $\sigma(p, q)$ over all $(p, q) \in Z_{\text{ver}}$ sum up to at most W. Thus if we can show that there is a horizontal line segment h such that the length of h equals the width of $\sigma(p, q)$ and $\mathcal{V} \cup \{h\}$ satisfies (p, q), then we are done.

Now observe that no line segment in $\mathcal{V}$ intersects $\sigma(p, q)$ since $\mathcal{V}$ is nice and $\sigma(p, q) \cap P = \emptyset$. Hence the segments of $\mathcal{V}$ that intersect R in fact intersect only the vertical edges of R. We assume w.l.o.g. that $x_p < x_q$ and $y_p < y_q$ (otherwise rename and/or mirror P at the x-axis). This means that due to the definition of Z_{ver}, there is no input point vertically above p. Thus if there is a segment s_p in $\mathcal{V}$ that intersects the left edge of R, then s_p must contain p. Analoguously, a segment s_q in $\mathcal{V}$ that intersects the right edge of R must contain q. Since $\mathcal{V}$ covers R, s_p or s_q must exist. Let ℓ be the horizontal through the topmost point of s_p or the bottommost point of s_q. Then $h = \ell \cap R$ does the job, again due to the fact that $\mathcal{V}$ covers R. Clearly h can be determined in constant time. □

In order to see that every point set has in fact a nice MVC, we need the following definitions. We restrict ourselves to the vertical case.

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 v is contained in a degenerate rectangle or if the number of points to the left of v that belong to the same connected component of G_ℓ is odd, otherwise we say that v is *even*. For a vertical line g let an *odd segment* be an inclusion-maximal connected set of odd points on g. Define *even segments* accordingly. For example, the segment s (drawn bold in Figure 4) is an even segment, while $f \setminus s$ is odd. We say that *parity changes* in points where two segments of different parity touch. We refer to these points as *points of changing parity.* The MVC with the desired property will simply be the set of all odd segments. The next lemma characterizes odd segments. Strictly speaking we have to state the parity of segment endpoints, but since a closed segment has the same length as the corresponding open segment, we consider odd segments closed.

Lemma 4. *Let $g : x = x_g$ be a vertical line through some $p = (x_p, y_p) \in P$.*

(i) Let e be a vertical edge of a rectangle $R \in \mathcal{R}_{\text{ver}}$. Then either all points on e are even or the only inclusion-maximal connected set of odd points on e contains an input point.
(ii) Let $R_1, \dots, R_d$ and $R'_1, \dots, R'_{d'}$ be the degenerate and non-degenerate rectangles in $\mathcal{R}_{\text{ver}}$ that g intersects, respectively. Then $d = |g \cap P| - 1$ and $d' \leq 2$. If $d = 0$ then $d' > 0$ and each R'_i has a corner in p. Else, if $d > 0$, there are $p_1, p_2 \in P$ such that $g \cap (R_1 \cup \dots \cup R_d) = \text{Seg}[p_1, p_2]$. Then each R'_i has a corner in either p_1 or p_2.
(iii) There are $b_g < t_g \in \mathbb{R}$ such that $g \cap \mathcal{A}_{\text{ver}} = \{x_g\} \times [b_g, t_g]$.
(iv) The line g contains at most two points of changing parity and at most one odd segment. For each point c of changing parity there is an input point with the same y-coordinate.

Proof. For (i) we assume without loss of generality that e is the right vertical edge of $R = \text{BBox}(p, q)$ and that q is the topmost point of e. If R is degenerate it is clear that all points on e (including p and q) are odd, and we are done. Thus we can assume that $x_p < x_q$. Let $p_0 = q, p_1 = p, p_2 \dots, p_k$ 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 $p_i = (x_i, y_i)$. For $2 \leq i \leq k$ 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 $p_0, p_1, \overline{p_2}, \overline{p_3}, \dots, \overline{p_k}$, see Figure 4. Note that the parity of a point v on e is determined by the number of segments of $\overline{\mathcal{L}}$ that the horizontal h_v through v intersects. If h_v 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_v 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. In other words, if $\overline{y_k} = y_0$, all points of e are even, if $\overline{y_k} = y_1$, all points of e are odd, and otherwise parity changes only in $(x_0, \overline{y_k})$ and q is odd. This settles (i).

(ii) follows directly from the definition of Z_{ver}, and (iii) follows from (ii).

For (iv) we first assume $d = 0$. Then (ii) yields $d' \in \{1, 2\}$ and $g \cap P = \{p\}$. By (i) we know that the only inclusion-maximal connected set of odd points on each vertical rectangle edge on g contains an input point, i.e. p. Thus there are at most two points of changing parity and at most one odd segment on g. Also according to the above proof of (ii), parity can change only in points of type $(x_0, \overline{y_k})$, and $\overline{y_k}$ is the y-coordinate of some input point in the set $\{p_0, \dots, p_k\}$.

Now if $d > 0$ note that all degenerate rectangles consist only of odd points. By (ii) we have that $g \cap (R_1 \cup \dots \cup R_d) = \text{Seg}[p_1, p_2]$ and that each of the at most two non-degenerate rectangles has a corner in either p_1 or p_2. Thus again the statement holds. □

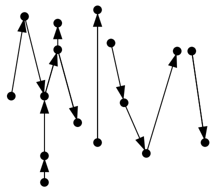
Fig. 1. Point pairs in Z_{ver}

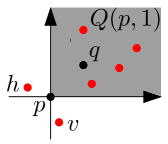

Fig. 2. The pair (p, q) is in Z_{quad}

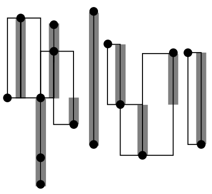
Fig. 3. The odd MVC

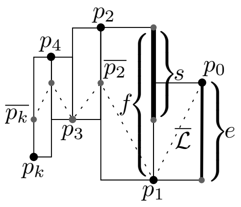

Fig. 4. Proof of Lemma 4

Lemma 5. *The set $\mathcal{V}$ of all odd segments is a nice* MVC, *the* odd MVC.

Proof. Clearly $\mathcal{V}$ covers $\mathcal{R}_{\text{ver}}$. Let ℓ be a horizontal line 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 $(k-1)/2$. If $k = 1$ any cover must contain the vertex, and so does $\mathcal{V}$ since the vertex belongs to a degenerate rectangle. Thus $\mathcal{V}$ is an MVC. Lemma 4 (i) shows that $\mathcal{V}$ is nice. □

Lemma 6. *The odd* MVC *can be computed in $O(n \log n)$ time using linear space.*

To compute the odd MVC we sweep the plane from bottom to top. For each point c of changing parity there is an input point p with $y_c = y_p$. Thus, the event-point queue can be implemented as a sorted list of the y-coordinates of the input points. The sweep-line status is a balanced binary tree in which each node corresponds to a connected components of G_ℓ, where ℓ is the current position of the horizontal sweep line. For details we refer to the full version of this paper [2].

4 An Approximation Algorithm

Our algorithm ApproxMMN proceeds in four phases, see Figure 5.

Phase 0. In phase 0 we compute all pointers of types xnbor and ynbor, and the set Z. From now on we will organize our data structures such that we have constant-time access to all relevant information such as xnbor, ynbor, vertical and horizontal predecessors and successors from each point $p \in P$.

Phase I. First we compute the nice odd MVC and the nice odd MHC, denoted by $\mathcal{C}_{\text{ver}}$ and $\mathcal{C}_{\text{hor}}$, respectively. Then we compute the set $\mathcal{S}$ of additional segments according to Lemma 3. We compute $\mathcal{C}_{\text{ver}}$, $\mathcal{C}_{\text{hor}}$ and $\mathcal{S}$ such that from each point $p \in P$ we have constant-time access to the at most two *cover segments* (i.e. segments in $\mathcal{C}_{\text{ver}} \cup \mathcal{C}_{\text{hor}}$) that contain p and to the at most four additional segments in rectangles incident to p.

Lemmas 1, 3, and 6 show that $N_1 = \mathcal{C}_{\text{ver}} \cup \mathcal{C}_{\text{hor}} \cup \mathcal{S}$ can be computed in $O(n \log n)$ time and that $|N_1| \leq |N_{\text{opt}}| + H + W$ holds. Recall that N_{opt} is a fixed MMN.

Phase II. In general N_1 does not satisfy Z_{quad}; further segments are needed. In order to be able to bound the length of these new segments, we partition the plane into two areas $\mathcal{A}_{12}$ and $\mathcal{A}_3$ as indicated in Section 2. We wanted to define $\mathcal{A}_3$ such that $|N_{\text{opt}} \cap \mathcal{A}_3|$ were large enough for us to bound the length of the new segments. However, we were not able to define $\mathcal{A}_3$ such that we could at the same time (a) satisfy Z_{quad} by adding new segments exclusively in $\mathcal{A}_3$ and (b) bound their length. Therefore we put the new segments into two disjoint sets, N_2 and N_3, such that $N_1 \cup N_2 \subseteq \mathcal{A}_{12}$ and $N_3 \subseteq \mathcal{A}_3$. This enabled us to bound $|N_1 \cup N_2|$ by $3|N_{\text{opt}} \cap \mathcal{A}_{12}|$ and $|N_3|$ by $2|N_{\text{opt}} \cap \mathcal{A}_3|$.

We now prepare our definition of $\mathcal{A}_3$. Recall that $Q(q,1), \ldots, Q(q,4)$ are the four quadrants of the Cartesian coordinate system with origin q. Let $P(q,t) = \{p \in P \cap Q(q,t) \mid (p,q) \in Z_{\text{quad}}\}$ for $t = 1,2,3,4$. For example, in Figure 7, $P(q,1) = \{p_1, \ldots, p_5\}$. Due to the definition of Z_{quad} we have $Q(p,t) \cap P(q,t) = \{p\}$ for each $p \in P(q,t)$. Thus the area $\mathcal{A}_{\text{quad}}(q,t) = \bigcup_{p \in P(q,t)} \text{BBox}(p,q)$ is a staircase polygon. The points in $P(q,t)$ are the "stairs" of the polygon and q is the corner opposite the stairs. In Figure 7, $\mathcal{A}_{\text{quad}}(q,1)$ is the union of the shaded areas. In order to arrive at a definition of the area $\mathcal{A}_3$, we will start from polygons of type $\mathcal{A}_{\text{quad}}(q,t)$ and then subtract areas that can contain segments of N_1 or are not needed to satisfy Z_{quad}.

Let $\mathit{\Delta}(q,t) = \text{int}\big(\mathcal{A}_{\text{quad}}(q,t) \setminus (\mathcal{A}_{\text{hor}} \cup \mathcal{A}_{\text{ver}})\big)$, where $\text{int}(M)$ denotes the interior of a set $M \subseteq \mathbb{R}^2$. In Figure 7, $\mathit{\Delta}(q,1)$ is the union of the three areas with dotted boundary. Let $\delta(q,t)$ be the union of those connected components A of $\mathit{\Delta}(q,t)$, such that $\partial A \cap P(q,t) \neq \emptyset$. In Figure 7, $\delta(q,1)$ is the union of the two dark shaded areas A and $\overline{A}$.

Due to the way we derived $\delta(q,t)$ from $\mathcal{A}_{\text{quad}}(q,t)$, it is clear that each connected component A of $\delta(q,t)$ is a staircase polygon, too. The stairs of A correspond to the input points on ∂A, i.e. $P(q,t) \cap \partial A$. Let q_A be the point on ∂A that is closest to q. This is the corner of A opposite the stairs. The next lemma, which is proved in [2], is crucial for estimating the length of our network within the $\delta(q,t)$ regions.

Lemma 7. *Areas of type $\delta(q,t)$ are pairwise disjoint.*

By Lemma 7 we are sure that we can treat each connected component A of $\delta(q,t)$ independently. Finally we define $\mathcal{A}_3 = \bigcup_{t \in \{1,2,3,4\}} \bigcup_{q \in P} \delta(q,t)$ and $\mathcal{A}_{12} = \mathbb{R}^2 \setminus \mathcal{A}_3$. This definition ensures that $N_1 \subset \mathcal{A}_{12}$ as desired. The set N_2 will be constructed as follows: for each connected component A of $\mathcal{A}_3$, we put $\partial A \setminus \bigcup N_1$ into N_2 and test whether N_1 contains a Manhattan path from q_A to q. If not, we add a further segment to N_2. This segment lies in $\mathcal{A}_{\text{hor}}$ and will be defined below. Since $\mathcal{A}_{\text{hor}}$ as well as ∂A are contained in $\mathcal{A}_{12}$, we have $N_2 \subset \mathcal{A}_{12}$. The set N_3 will be defined in phase III and will be arranged such that $N_3 \subset \mathcal{A}_3$.

We now describe how to compute $P(q,t)$ and how to find the connected components of $\delta(q,t)$. We compute all sets $P(q,t)$ by going through the input points and checking their Z_{quad}-partners. This takes linear time since $|Z_{\text{quad}}| = O(n)$. We sort the points in each set $P(q,t)$ according to their x-distance from q. This takes $O(n \log n)$ total time. The remaining difficulty is to decide which points in $P(q,t)$ are incident to the same connected component of $\delta(q,t)$. In Figure 7, $\{p_1, p_2\} \subset \partial A$ and $\{p_3, p_4, p_5\} \subset \partial \overline{A}$. For our description how to figure this out we assume $t = 1$ and $P(q,1) = (p_1, \dots, p_m)$. Note that each connected component of $\delta(q,1)$ corresponds to a sequence of consecutive points in $P(q,1)$. By definition, for each connected component A of $\delta(q,1)$ and all $p_i, p_j \in A$ we have $p_i.\text{ynbor}[3] = p_j.\text{ynbor}[3]$.

We detect these sequences by going through $p_1, \dots, p_m$. Let p_i be the current point and let A be the current connected component. If and only if $p_i.\text{ynbor}[3] \neq p_{i+1}.\text{ynbor}[3]$ there is a rectangle $R_A \in \mathcal{R}_{\text{hor}}$ that separates A from the next connected component of $\delta(q,1)$. The rectangle R_A is defined by the point $v_A = p_i.\text{ynbor}[3]$ and its horizontal successor w_A, which in this case is unique, see Figure 7. It remains to specify the coordinates of the corner point q_A of A. Let p_0 be the (unique) vertical successor of q. Then $x_{q_A} = x_{p_0}$ and $y_{q_A} = y_{w_A}$.

At last, we want to make sure that $N_1 \cup N_2$ contains a Manhattan q–q_A path. The reason for this is that in phase III we will only compute Manhattan paths from each $p_i \in \partial A$ to q_A. Concatenating these paths with the q–q_A path yields Manhattan p_i–q paths since $q_A \in \text{BBox}(q, p_i)$. Note that segments in N_3 lie in $\mathcal{A}_3$ and thus cannot help to establish a q–q_A path within $\text{BBox}(q, q_A) \subset \mathcal{A}_{12}$.

The set N_1 contains a Manhattan q–p_0 path $\mathcal{P}_{\text{ver}}$ and a Manhattan v_A–w_A path $\mathcal{P}_{\text{hor}}$, since $(q, p_0) \in Z_{\text{ver}}$ and $(v_A, w_A) \in Z_{\text{hor}}$. If $q_A \in \mathcal{P}_{\text{ver}}$, then clearly N_1 contains a Manhattan q–q_A path. However, N_1 also contains a Manhattan q–q_A path if $q_A \in \mathcal{P}_{\text{hor}}$. This is due to the fact that $\mathcal{P}_{\text{ver}}$ and $\mathcal{P}_{\text{hor}}$ intersect. If $q_A \notin \mathcal{P}_{\text{ver}} \cup \mathcal{P}_{\text{hor}}$, then $\mathcal{P}_{\text{hor}}$ contains the point $c_A = (x_{q_A}, y_{v_A})$, which lies on the vertical through q_A on the opposite edge of R_A. Thus, to ensure a Manhattan q–q_A path in $N_1 \cup N_2$, it is enough to add the segment $s_A = \text{Seg}[q_A, c_A]$ to N_2. We refer to such segments as *connecting segments.*

The algorithm ApproxMMN does not compute $\mathcal{P}_{\text{ver}}$ and $\mathcal{P}_{\text{hor}}$ explicitly, but simply tests whether $q_A \notin \bigcup N_1$. This is equivalent to $q_A \notin \mathcal{P}_{\text{ver}} \cup \mathcal{P}_{\text{hor}}$ since our covers are minimum and the bounding boxes of $\mathcal{P}_{\text{ver}}$ and $\mathcal{P}_{\text{hor}}$ are the only rectangles in $\mathcal{R}_{\text{ver}} \cup \mathcal{R}_{\text{hor}}$ that contain s_A. Due to the same reason and to the fact that cover edges are always contained in (the union of) edges of rectangles in $\mathcal{R}_{\text{ver}} \cup \mathcal{R}_{\text{hor}}$, we have that $s_A \cap \bigcup N_1 = \{c_A\}$. This shows that connecting segments intersect N_1 at most in endpoints. The same holds for segments in N_2 that lie on $\partial \mathcal{A}_3$. We summarize:

Lemma 8. *In $O(n \log n)$ time we can compute the set N_2, which has the following properties: (i) $N_2 \subset \mathcal{A}_{12}$, (ii) a segment in N_1 and a segment in N_2 intersect at most in their endpoints, and (iii) for each region $\delta(q,t)$ and each connected component A of $\delta(q,t)$, $N_1 \cup N_2$ contains ∂A and a Manhattan q–q_A path.*

Proof. The properties of N_2 follow from the description above. The runtime is as follows. Let A be a connected component of $\mathcal{A}_3$ and $m_A = |P \cap \partial A|$. Note that $\sum m_A = O(n)$ since each point is adjacent to at most four connected components

```
ApproxMMN($P$)

Phase 0:  Neighbors and generat. set.
for each $p \in P$ and $t \in \{1,2,3,4\}$ do
    compute $p$.xnbor[$t$] and $p$.ynbor[$t$]
compute $Z = Z_{\text{ver}} \cup Z_{\text{hor}} \cup Z_{\text{quad}}$.

Phase I:  Compute $N_1$.
compute odd MVC $\mathcal{C}_{\text{ver}}$ and MHC $\mathcal{C}_{\text{hor}}$
compute set $\mathcal{S}$ of additional segments
$N_1 \leftarrow \mathcal{C}_{\text{ver}} \cup \mathcal{C}_{\text{hor}} \cup \mathcal{S},\ N_2 \leftarrow \emptyset,\ N_3 \leftarrow \emptyset$

Phase II:  Compute $N_2$.
compute $\mathcal{A}_3$
for each connected comp. $A$ of $\mathcal{A}_3$ do
    $N_2 \leftarrow N_2 \cup (\partial A \setminus \bigcup N_1)$
    if $q_A \notin \bigcup N_1$ then
        $N_2 \leftarrow N_2 \cup \{s_A\}$

Phase III: Compute $N_3$.
for each connected comp. $A$ of $\mathcal{A}_3$ do
    $N_3 \leftarrow N_3 \cup$ Bridge($A$)

return $N = N_1 \cup N_2 \cup N_3$
```

Fig. 5.

```
Bridge($A = (q_A, p_1, \ldots, p_m)$)

for $i = 1$ to $m-1$ do
    compute $\alpha_i$ and $\beta_i$
return SubBridge($1, m, 0, 0$)

SubBridge($k, l, x_{\text{off}}, y_{\text{off}}$)

$A_{\text{curr}} = (q_A + (x_{\text{off}}, y_{\text{off}}), p_k, \ldots, p_l)$
if $l - k < 2$ return $\emptyset$
$\Lambda = \{j \in \{k, \ldots, l-1\}:$
        $\alpha_j - x_{\text{off}} \le \beta_j - y_{\text{off}}\}$
$i = \max \Lambda \cup \{k\}$
if $i < l-1$ and $\alpha_i - x_{\text{off}} \le \beta_{i+1} - y_{\text{off}}$
    then $i = i + 1$
$B = \emptyset$
if $i > 1$      then $B = B \cup \{a_{i-1} \cap A_{\text{curr}}\}$
if $i < l-1$ then $B = B \cup \{b_{i+1} \cap A_{\text{curr}}\}$
$x_{\text{new}} = x(p_{i+1}) - x(q_A)$
$y_{\text{new}} = y(p_i) - y(q_A)$
return $B \cup$
  $\cup$ SubBridge($l, i-1, x_{\text{off}}, y_{\text{new}}$)
  $\cup$ SubBridge($i+2, l, x_{\text{new}}, y_{\text{off}}$)
```

Fig. 6.

of $\mathcal{A}_3$, according to Lemma 7. After sorting $P(q,t)$ we can compute in $O(m)$ time for each A the segment s_A and the set $\partial A \setminus \bigcup N_1$. This is due to the fact that we have constant-time access to each of the $O(m)$ rectangles in $\mathcal{R}_{\text{hor}} \cup \mathcal{R}_{\text{ver}}$ that intersect ∂A and to the $O(m)$ segments of N_1 that lie in these rectangles. $\square$

Phase III. Now we finally satisfy the pairs in Z_{quad}. Due to Lemma 8 it is enough to compute, for each connected component A of $\mathcal{A}_3$, a set of segments $B(A)$ such that the union of $B(A)$ and ∂A contains Manhattan paths from any input point on ∂A to q_A. We say that such a set $B(A)$ *bridges* A. The set N_3 will be the union over all sets of type $B(A)$. The algorithm Bridge that we use to compute $B(A)$ is similar to the "thickest-first" greedy algorithm for rectangulating staircase polygons, see [3]. However, we cannot use that algorithm since the segments that it computes do not lie entirely in $\mathcal{A}_3$.

For our description of algorithm Bridge, and also in the pseudocode in Figure 6, we assume that A lies in a region of type $\delta(q,1)$. Let again $(p_1, \ldots, p_m)$ denote the sorted sequence of points on ∂A. Note that ∂A already contains Manhattan paths that connect p_1 and p_m to q_A. Thus we are done if $m \le 2$. Otherwise let $p'_j = (x_{p_j}, y_{p_{j+1}})$, $a_j = \text{Seg}[(x_{q_A}, y_{p'_j}), p'_j]$ and $b_j = \text{Seg}[(x_{p'_j}, y_{q_A}), p'_j]$ for $j \in \{1, \ldots, m-1\}$, see Figure 8. We denote $|a_j|$ by α_j and $|b_j|$ by β_j. From now on we identify staircase polygon A with the tupel $(q_A, p_1, \ldots, p_m)$. Let B be the set of segments that algorithm Bridge computes. Initially $B = \emptyset$. The algorithm chooses an $i \in \{1, \ldots, m-1\}$ and adds—if they exist—a_{i-1} and b_{i+1} to B.

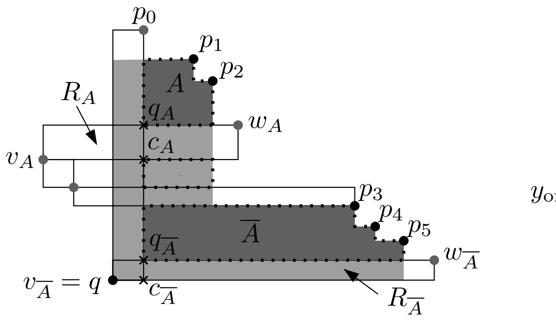

Fig. 7. Notation: $\mathcal{A}_{\text{quad}}(q,1)$ shaded, $\Delta(q,1)$ with dotted boundary, and $\delta(q,1) = A \cup A'$ dark shaded

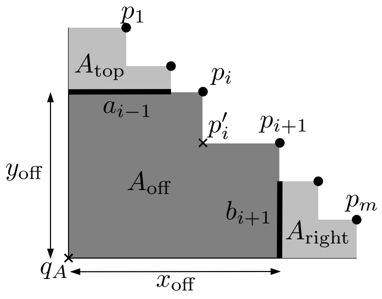

Fig. 8. Notation for algorithm Bridge

This satisfies $\{(p_i, q), (p_{i+1}, q)\}$. In order to satisfy $\{(p_2, q), \dots, (p_{i-1}, q)\}$ and $\{(p_{i+2}, q), \dots, (p_{m-1}, q)\}$, we solve the problem recursively for the two staircase polygons $((x_{q_A}, y_{p_i}), p_1, \dots, p_{i-1})$ and $((x_{p_{i+1}}, y_{q_A}), p_{i+2}, \dots, p_m)$.

Our choice of i is as follows. Note that $\alpha_1 < \dots < \alpha_{m-1}$ and $\beta_1 > \dots > \beta_{m-1}$. Let $\Lambda = \{j \in \{1, \dots, m-1\} \mid \alpha_j \le \beta_j\}$. If $\Lambda = \emptyset$, we have $\alpha_1 > \beta_1$, i.e. A is flat and broad. In this case we choose $i = 1$, which means that only b_2 is put into B. Otherwise let $i' = \max \Lambda$. Now if $i' < m-1$ and $\alpha_{i'} \le \beta_{i'+1}$, we choose $i = i' + 1$. In all other cases let $i = i'$. The idea behind this choice of i is that it yields a way to balance α_{i-1} and β_{i+1}, which in turn helps to compare $\alpha_{i-1} + \beta_{i+1}$ to $\min\{\alpha_i, \beta_i, \alpha_{i-1} + \beta_{i+1}\}$, i.e. the length of the segments needed by any Manhattan network to connect p_i and p_{i+1} to q.

To avoid expensive updates of the α- and β-values of the staircase polygons in the recursion, we introduce offset values x_{off} and y_{off} that denote the x-respectively y-distance from the corner of the current staircase polygon to the corner q_A of A. In order to find the index i in a recursion, we compare $\alpha_j - x_{\text{off}}$ to $\beta_j - y_{\text{off}}$ instead of α_j to β_j as in the definition of Λ above.

Running time and performance of algorithm Bridge are as follows:

Theorem 1. *Given a connected component A of $\mathcal{A}_3$ with $|P \cap \partial A| = m$, algorithm* Bridge *computes in $O(m \log m)$ time a set B of line segments with $|B| \le 2|N_{\text{opt}} \cap A|$ and $\bigcup B \subset A$ that bridges A.*

Our proof of Theorem 1 is similar to the analysis of the greedy algorithm for rectangulation, see Theorem 10 in [3]. The details can again be found in [2].

We conclude this section by analyzing the running time of ApproxMMN.

Theorem 2. ApproxMMN *runs in $O(n \log n)$ time and uses $O(n)$ space.*

Proof. Each of the four phases of our algorithm takes $O(n \log n)$ time: for phase 0 refer to Lemma 2, for phase I to Lemmas 3 and 6, for phase II to Lemma 8 and for phase III to Theorem 1. ApproxMMN outputs $O(n)$ line segments. □

5 The Approximation Factor

As desired we can now bound the length of N in $\mathcal{A}_{12}$ and $\mathcal{A}_3$ separately. Theorem 1 and Lemma 7 directly imply that $|N \cap \mathcal{A}_3| = |N_3| \leq 2|N_{\text{opt}} \cap \mathcal{A}_3|$. Note that by $|N_{\text{opt}} \cap \mathcal{A}_3|$ we actually mean $|\{s \cap \mathcal{A}_3 : s \in N_{\text{opt}}\}|$. It remains to show that $|N \cap \mathcal{A}_{12}| = |N_1 \cup N_2|$ is bounded by $3|N_{\text{opt}} \cap \mathcal{A}_{12}|$.

Recall that by Lemmas 1 and 3, $|N_1| \leq |N_{\text{opt}}| + H + W$. Since the segments of N_{opt} that were used to derive the estimation of Lemma 1 lie in $\mathcal{A}_{\text{ver}} \cup \mathcal{A}_{\text{hor}} \subset \mathcal{A}_{12}$, even the stronger bound $|N_1| \leq |N_{\text{opt}} \cap \mathcal{A}_{12}| + H + W$ holds. It remains to analyze the length of N_2 segments. Let N_2^{ver} (N_2^{hor}) denote the set of all vertical (horizontal) segments in N_2. We call segments of N_2 that lie on $\partial\mathcal{A}_3$ *boundary segments*. Due to Lemma 8, segments in N_2^{ver} and segments in $\mathcal{C}_{\text{ver}}$ intersect at most in segment endpoints. Thus, a horizontal line ℓ with $\ell \cap P = \emptyset$ does not contain any point that lies at the same time in $\bigcup \mathcal{C}_{\text{ver}}$ and in $\bigcup N_2^{\text{ver}}$. In the full version [2] we characterize the sequences that are obtained by the intersection of such a line ℓ with cover segments, boundary segments, and connecting segments. A counting argument then yields $\#N_2^{\text{ver}} \leq 2\#\mathcal{C}_{\text{ver}} - 1$ (and analogously $\#N_2^{\text{hor}} \leq 2\#\mathcal{C}_{\text{hor}} - 1$), where $\#N_2^{\text{ver}}$ and $\#\mathcal{C}_{\text{ver}}$ denote the number of segments in N_2^{ver} and $\mathcal{C}_{\text{ver}}$ intersected by ℓ, respectively. By integrating this inequality over all positions of ℓ, we obtain the following lemma.

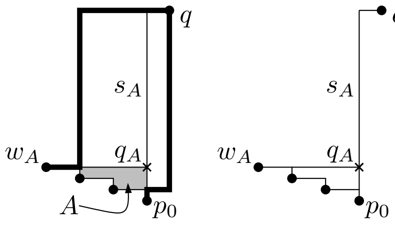

Fig. 9. Left: In the network N_1 neither the w_A–q nor the p_0–q path (thick) contain q_A. Thus $s_A \in N$. Right: MMN of the same point set.

Lemma 9. $|N_2^{\text{ver}}| \leq 2|\mathcal{C}_{\text{ver}}| - H$ *and* $|N_2^{\text{hor}}| \leq 2|\mathcal{C}_{\text{hor}}| - W$.

This finally settles the approximation factor of ApproxMMN.

Theorem 3. $|N| \leq 3|N_{\text{opt}}|$.

Proof. By Lemma 9 and $|\mathcal{C}_{\text{ver}} \cup \mathcal{C}_{\text{hor}}| \leq |N_{\text{opt}} \cap \mathcal{A}_{12}|$ we have $|N_2| \leq 2|N_{\text{opt}} \cap \mathcal{A}_{12}| - H - W$. Together with $|N_1| \leq |N_{\text{opt}}| + H + W$ this yields $|N_1 \cup N_2| / |N_{\text{opt}} \cap \mathcal{A}_{12}| \leq 3$. Theorem 1 and Lemma 7 show that $|N_3| / |N_{\text{opt}} \cap \mathcal{A}_3| \leq 2$. Then, $\mathcal{A}_{12} \cap \mathcal{A}_3 = \emptyset$ yields $|N| / |N_{\text{opt}}| \leq \max\{|N_1 \cup N_2| / |N_{\text{opt}} \cap \mathcal{A}_{12}|, |N_3| / |N_{\text{opt}} \cap \mathcal{A}_3|\} \leq 3$. $\square$

Figure 9 shows that there are point sets for which $|N|/|N_{\text{opt}}|$ can be made arbitrarily close to 3. However, an experimental evaluation of ApproxMMN shows that it behaves much better on point sets under various random distributions. The average performance was around 1.2. Details can be found in [2]. Our algorithm can be tested under the URL `http://i11www.ira.uka.de/manhattan/`. We close with the obvious question: is it NP-hard to compute an MMN?

References

1. 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. (STOC'95)*, pages 489–498, Las Vegas, 29 May–1 June 1995.
2. M. Benkert, F. Widmann, and A. Wolff. The minimum Manhattan network problem: A fast factor-3 approximation. Technical Report 2004-16, Fakultät für Informatik, Universität Karlsruhe, 2004. Available at `http://www.ubka.uni-karlsruhe.de/cgi-bin/psview?document=/ira/2004/16`.
3. J. Gudmundsson, C. Levcopoulos, and G. Narasimhan. Approximating a minimum Manhattan network. *Nordic J. Comput.*, 8:219–232, 2001.
4. R. Kato, K. Imai, and T. Asano. An improved algorithm for the minimum Manhattan network problem. In P. Bose and P. Morin, editors, *Proc. 13th Int. Symp. Alg. and Comp. (ISAAC'02)*, vol. 2518 of *LNCS*, pp. 344–356, 2002. Springer-Verlag.
5. F. Lam, M. Alexandersson, and L. Pachter. Picking alignments from (Steiner) trees. *Journal of Computational Biology*, 10:509–520, 2003.