Approximating Minimum Manhattan Networks in Higher Dimensions

Aparna Das[1], Emden R. Gansner[2], Michael Kaufmann[3], Stephen Kobourov[1], Joachim Spoerhase[4], and Alexander Wolff[4]

[1] Dept. of Comp. Sci., University of Arizona, Tucson, AZ, U.S.A.
[2] AT&T Labs Research, Florham Park, NJ, U.S.A.
[3] Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, Germany
[4] Institut für Informatik, Universität Würzburg, Germany

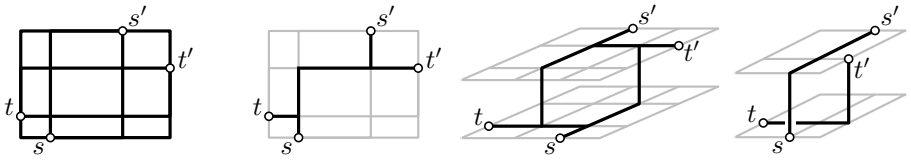
Abstract. We consider the minimum Manhattan network problem, which is defined as follows. Given a set of points called *terminals* in $\mathbb{R}^d$, find a minimum-length network such that each pair of terminals is connected by a set of axis-parallel line segments whose total length is equal to the pair's Manhattan (that is, L_1-) distance. The problem is NP-hard in 2D and there is no PTAS for 3D (unless $\mathcal{P} = \mathcal{NP}$). Approximation algorithms are known for 2D, but not for 3D.
We present, for any fixed dimension d and any $\varepsilon > 0$, an $O(n^\varepsilon)$-approximation. For 3D, we also give a $4(k-1)$-approximation for the case that the terminals are contained in the union of $k \geq 2$ parallel planes.

1 Introduction

In a typical network construction problem, one is given a set of objects to be interconnected such that some constraints regarding the connections are fulfilled. Additionally, the network must be cheap. For example, if the objects are points in Euclidean space and the constraints say that, for some fixed $t > 1$, each pair of points must be connected by a path whose length is bounded by t times the Euclidean distance of the points, then the solution is a so-called *Euclidean t-spanner*. Concerning cost, one usually requires that the total length of the network is proportional to the length of a Euclidean minimum spanning tree of the points. Such cheap spanners can be constructed efficiently [2].

In this paper, we are interested in constructing *1*-spanners, with respect to the Manhattan (or L_1-) metric. Our aim is to minimize the total length (or *weight*) of the network. Note that the Euclidean 1-spanner of a set of points is simply the complete graph (if no three points are collinear) and hence, its weight is completely determined. Manhattan 1-spanners, in contrast, have many degrees of freedom and vastly different weights.

More formally, given two points p and q in d-dimensional space $\mathbb{R}^d$, a *Manhattan path* connecting p and q (a *p–q M-path*, for short) is a sequence of axis-parallel line segments connecting p and q whose total length equals the Manhattan distance of p and q. Thus an M-path is a monotone rectilinear path. For our purposes, a set of axis-parallel line segments is a *network*. Given a network N,



(a) an M-network for $T = \{s, s', t, t'\}$ (b) a minimum M-network for T (c) a minimum M-network in 3D (d) M-paths missing each other

Fig. 1: Examples of M-networks in 2D and 3D.

its *weight* $\|N\|$ is the sum over the lengths of its line segments. A network N *Manhattan-connects* (or *M-connects*) two given points p and q if it "contains" a p–q M-path π. Note that we slightly abuse the notation here: we mean pointwise containment, that is, we require $\bigcup \pi \subseteq \bigcup N$. Given a set T of points—called *terminals*—in $\mathbb{R}^d$, a network N is a *Manhattan network* (or *M-network*) for T if N M-connects every pair of terminals in T. The *minimum Manhattan network problem* (MMN) consists of finding, for a given set T of terminals, a minimum-weight M-network. For examples, see Fig. 1.

M-networks have important applications in computational biology; Lam et al. [13] use them in gene alignment in order to reduce the size of the search space of the Viterbi algorithm for pair hidden Markov models.

Previous work. 2D-MMN, the 2D-version of the problem, was introduced by Gudmundsson et al. [10]. They gave an 8- and a 4-approximation algorithm. Later, the approximation ratio was improved to 3 [3,9] and then to 2, which is the currently best result. It was achieved in three different ways: via linear programming [5], using the primal–dual scheme [16] and with purely geometric arguments [11]. The last two algorithms run in $O(n \log n)$ time. A ratio of 1.5 was claimed [17], but apparently the proof is incomplete [9]. Chin et al. [6] finally settled the complexity of 2D-MMN by proving it NP-hard.

A little earlier, Muñoz et al. [15] considered 3D-MMN. They showed that the problem is NP-hard and NP-hard to approximate beyond a factor of 1.00002. For the special case of 3D-MMN where any cuboid spanned by two terminals contains other terminals or is a rectangle, they gave a 2α-approximation, where α denotes the best approximation ratio for 2D-MMN. They posed the design of approximations algorithms for general 3D-MMN as an open problem.

Related problems. In d-dimensional MMN (dD-MMN) we consider the dimension d fixed. As we observe in the full version of our paper [7], dD-MMN is a special case of the *directed Steiner forest problem* (DSF). More precisely, an instance of MMN can be decomposed into a constant number of DSF instances. The input of DSF is an edge-weighted directed graph G and a set of vertex pairs. The goal is to find a minimum-cost subgraph of G (not necessarily a forest!) that connects all given vertex pairs. Recently, Feldman et al. [8] reported an $O(n^{4/5+\varepsilon})$-approximation for DSF. This bound carries over to dD-MMN.

An important special case of DSF is the *directed Steiner* tree *problem* (DST). Here, the input instance specifies a digraph G, a *root* vertex r, and a subset S of the vertices of G that must be connected to r. An optimum solution for DST is an r-rooted subtree of G spanning S. DST admits an $O(n^\varepsilon)$-approximation for any $\varepsilon > 0$ [4].

A *geometric* optimization problem that resembles MMN is the *rectilinear Steiner arborescence problem* (RSA). Given a set of points in $\mathbb{R}^d$ with non-negative coordinates, a rectilinear Steiner arborescence is a spanning tree that connects all points with M-paths to the origin. As for MMN, the aim is to find a minimum-weight network. For 2D-RSA, there is a polynomial-time approximation scheme (PTAS) [14] based on Arora's technique [1]. It is not known whether 2D-MMN admits a PTAS. Arora's technique does not seem to work here as M-paths between terminals forbid detours and thus may not respect portals. RSA is a special case of DST; see the full version of this paper [7].

Our contribution. We give, for any $\varepsilon > 0$, an $O(n^\varepsilon)$-approximation algorithm for 3D-MMN; see Section 3. In the full version of this paper [7], we extend this result to arbitrary but constant dimension. We also present a $4(k-1)$-approximation algorithm for the special case of 3D-MMN where the given terminals are contained in $k \geq 2$ planes parallel to the x–y plane; see Section 2.

Our $O(n^\varepsilon)$-approximation for 3D-MMN constitutes a significant improvement upon the best known ratio of $O(n^{4/5+\varepsilon})$ for (general) directed Steiner forest [8]. We obtain this result by exploiting the geometric structure of the problem. To underline the relevance of our result, we remark that the bound of $O(n^\varepsilon)$ is the best known result also for other directed Steiner-type problems such as DST [4] or even acyclic DST [19].

Our $O(k)$-approximation for the k-plane case strongly relies on a deep result that Soto and Telha [18] achieved recently. They showed that, given a set of red and blue points in the plane, one can determine efficiently a minimum-cardinality set of points that together *pierce* all rectangles having a red point in the lower left corner and a blue point in the upper right corner. Their algorithm, together with an approximation algorithm for 2D-MMN, basically yields an approximation for the 2-plane case. We show how to generalize this idea to k planes.

Intuitively, what makes 3D-MMN more difficult than 2D-MMN is the fact that in 2D, if the bounding box of terminals s and s' and the bounding box of t and t' cross (as in Fig. 1b), then any s–s' M-path will intersect any t–t' M-path, which yields s–t' and t–s' M-paths for free (if s and t are the lower left corners of their respective boxes). A similar statement for 3D does not hold; M-paths can "miss" each other—even if their bounding cuboids cross; see Fig. 1d.

Let us formalize this observation. Given a set T of terminals, a set Z of pairs of terminals is a *generating set* [12] if any network that M-connects the pairs in Z in fact M-connects *all* pairs of terminals. In 2D, any MMN instance has a generating set of linear size [12]. This fact is exploited by most approximation algorithms for 2D-MMN. In the full version of this paper [7], we show that there are 3D instances where any generating set has quadratic size.

Notation and an observation. Given a point $p \in \mathbb{R}^3$, we denote the x-, y- and z-coordinate of p by $x(p)$, $y(p)$, and $z(p)$, respectively. Given two points a and c in $\mathbb{R}^2$, let $R(a,c) = \{b \in \mathbb{R}^2 \mid x(a) \leq x(b) \leq x(c), y(a) \leq y(b) \leq y(c)\}$ be the *rectangle spanned by a and c*. If a line segment is parallel to the x-, y-, or z-axis, we say that it is x-, y-, or z-*aligned*.

Gudmundsson et al. [10] observed that any instance of MMN has a solution that is contained in the *Hanan grid*, the grid induced by the terminals.

2 k-Plane Case

In this section, we consider 3D-MMN under the assumption that the set T of terminals is contained in the union of $k \geq 2$ planes $E_1, \ldots, E_k$ that are parallel to the x–y plane. Of course, this assumption always holds for some $k \leq n$. We present a $4(k-1)$-approximation algorithm, which outperforms our algorithm for the general case in Section 3 if $k \in o(n^\varepsilon)$.

Let N_{opt} be some fixed minimum M-network for T, let $N_{\text{opt}}^{\text{hor}}$ be the set of all x-aligned and all y-aligned segments in N_{opt}, and let $N_{\text{opt}}^{\text{ver}}$ be the set of all z-aligned segments in N_{opt}. Let OPT denote the weight of N_{opt}. Of course, OPT does not depend on the specific choice of N_{opt}; the weights of $N_{\text{opt}}^{\text{hor}}$ and $N_{\text{opt}}^{\text{ver}}$, however, may depend on N_{opt}. Further, let T_{xy} be the projection of T onto the x–y plane. For $i \in \{1, \ldots, k\}$, let $T_i = T \cap E_i$ be the set of terminals in plane E_i.

Our algorithm consists of two phases. Phase I computes a set N^{hor} of horizontal (that is, x- and y-aligned) line segments, phase II computes a set N^{ver} of vertical line segments. Finally, the algorithm returns the set $N = N^{\text{hor}} \cup N^{\text{ver}}$.

Phase I is simple; we compute a 2-approximate M-network N_{xy} for T_{xy} (using the algorithm of Guo et al. [11]) and project N_{xy} onto each of the planes $E_1, \ldots, E_k$. Let N^{hor} be the union of these projections. Note that N^{hor} M-connects any pair of terminals that lie in the same plane.

Observation 1 $\|N^{\text{hor}}\| \leq 2k\|N_{\text{opt}}^{\text{hor}}\|$.

Proof. The projection of $N_{\text{opt}}^{\text{hor}}$ to the x–y plane is an M-network for T_{xy}. Hence, $\|N_{xy}\| \leq 2N_{\text{opt}}^{\text{hor}}$. Adding up over the k planes yields the claim. □

In what follows, we describe phase II, which computes a set N^{ver} of vertical line segments, so-called *pillars*, of total cost at most $4(k-1)\|N_{\text{opt}}^{\text{ver}}\|$. This yields an overall approximation factor of $4(k-1)$ since $\|N^{\text{hor}} \cup N^{\text{ver}}\| \leq 2k\|N_{\text{opt}}^{\text{hor}}\| + 4(k-1)\|N_{\text{opt}}^{\text{ver}}\| \leq 4(k-1)(\|N_{\text{opt}}^{\text{hor}}\| + \|N_{\text{opt}}^{\text{ver}}\|) \leq 4(k-1)\text{OPT}$.

For simplicity, we restrict ourselves to the *directional* subproblem of M-connecting all terminal pairs (t, t') such that t *dominates* t', that is, $x(t) \leq x(t'), y(t) \leq y(t'), z(t) \leq z(t')$ and $t \neq t'$. We call such terminal pairs *relevant*.

One may think of this directional subproblem as solving the part of the problem that runs in direction north-east (NE) in the x–y plane (with increasing z-coordinates). We construct a pillar network $N_{\text{dir}}^{\text{ver}}$ of weight at most $(k-1)\|N_{\text{opt}}^{\text{ver}}\|$ that, together with N^{hor}, M-connects all relevant pairs. We solve the analogous subproblems for the directions NW, SE, and SW symmetrically. Then N^{ver} is

the union of the four partial solutions and has weight at most $4(k-1)\|N^{\text{ver}}_{\text{opt}}\|$ as desired.

Our directional subproblem is closely linked to the *(directional) bichromatic rectangle piercing problem* (BRP), which is defined as follows. Let R and B be sets of red and blue points in $\mathbb{R}^2$, respectively, and let $\mathcal{R}(R,B)$ denote the set of axis-aligned rectangles each of which is spanned by a red point in its SW-corner and a blue point in its NE-corner. Then the aim of BRP is to find a minimum-cardinality set $P \subset \mathbb{R}^2$ such that every rectangle in $\mathcal{R}(R,B)$ is *pierced*, that is, contains at least one point in P. The points in P are called *piercing points*.

The problem dual to BRP is the *(directional) bichromatic independent set of rectangles problem* (BIS) where the goal is to find the maximum number of pairwise disjoint rectangles in $\mathcal{R}(R,B)$, given the sets R and B.

Recently, Soto and Telha [18] proved a beautiful min–max theorem saying that, for $\mathcal{R}(R,B)$, the minimum number of piercing points always *equals* the maximum number of independent rectangles. This enabled them to give efficient exact algorithms for BRP and BIS running in $\tilde{O}(n^{2.5})$ worst-case time or $\tilde{O}(n^{\gamma})$ expected time, where the $\tilde{O}$-notation ignores polylogarithmic factors, $\gamma < 2.4$ is the exponent for fast matrix multiplication, and $n = |R| + |B|$ is the input size.

2.1 Two Planes

Our phase-II algorithm for two planes is very simple. We sketch it in order to give a smooth introduction to the k-plane case. Imagine the terminals in T_1 to be red and those in T_2 to be blue. Ignore the z-coordinates of the terminals. Then the relevant red–blue point pairs span exactly the rectangles in $\mathcal{R}(T_1,T_2)$, which we call relevant, too.

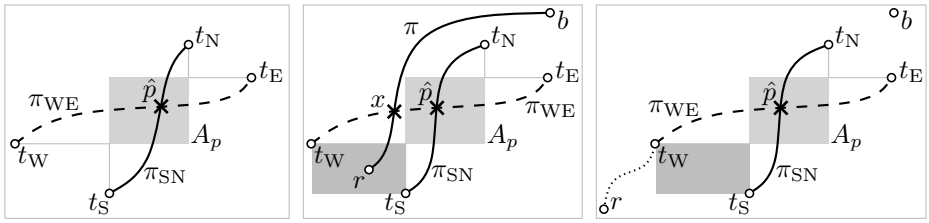
Our algorithm consists of two steps. First, we compute a minimum piercing P of $\mathcal{R}(T_1,T_2)$ using the algorithm of Soto and Telha [18]. Second, we move each piercing point $p \in P$ to a new position $\hat{p}$—a nearby junction of N_{xy}—and erect, at $\hat{p}$, a pillar connecting the two planes. Let $\hat{P}$ be the set of piercing points after the move, and let $N^{\text{ver}}_{\text{dir}}$ be the corresponding set of pillars.

Lemma 1. *It holds that* $\|N^{\text{ver}}_{\text{dir}}\| \le \|N^{\text{ver}}_{\text{opt}}\|$.

Proof. Clearly, $|\hat{P}| = |P|$. Integrating over the distance d of the two planes yields $\|N^{\text{ver}}_{\text{dir}}\| = |\hat{P}|\cdot d = |P|\cdot d \le \|N^{\text{ver}}_{\text{opt}}\|$. The last inequality is due to the fact that P is a *minimum* piercing of $\mathcal{R}(T_1,T_2)$ and that the pillars in $N^{\text{ver}}_{\text{opt}}$ pierce $\mathcal{R}(T_1,T_2)$—otherwise N_{opt} would not be feasible. □

Now we turn to feasibility. We first detail how we move each piercing point p to its new position $\hat{p}$. For the sake of brevity, we identify terminals with their projections to the x–y plane. Our description assumes that we have at our disposal some network M (such as N_{xy}) connecting the relevant pairs in T_{xy}.

Given a piercing point $p \in P$, let A_p be the intersection of the relevant rectangles pierced by p. Clearly, $p \in A_p$. Refer to Fig. 2a; note that the bottom and left sides of A_p are determined by terminals t_{W} and t_{S} to the west and south



(a) Paths π_{SN} and π_{WE} meet in a point $\hat{p}$ in A_p.

(b) Case I: we can go from r via x to $\hat{p}$.

(c) Case II: we can go from r via t_W or t_S to $\hat{p}$.

Fig. 2: Sketches for the proof of Lemma 2.

of A_p, respectively. Symmetrically, the top and right sides of A_p are determined by terminals t_E and t_N to the east and north of A_p, respectively. Note that t_W and t_S may coincide, and so may t_E and t_N. Clearly, the network M contains an M-path π_{SN} connecting t_S and t_N and an M-path π_{WE} connecting t_W and t_E. The path π_{SN} goes through the bottom and top sides of A_p and π_{WE} goes through the left and right sides. Hence, the two paths intersect in a point $\hat{p} \in A_p$. This is where we move the original piercing point p.

Since $\hat{p} \in A_p$, the point $\hat{p}$ pierces the same relevant rectangles as p, and the set $\hat{P} = \{\hat{p} \mid p \in P\}$ is a (minimum) piercing for the set of relevant rectangles.

Lemma 2. *Let $\mathcal{R}(R, B)$ be an instance of* BRP *and let M be a network that M-connects every relevant red–blue point pair. Then we can efficiently compute a minimum piercing of $\mathcal{R}(R, B)$ such that M contains, for every relevant red–blue point pair (r, b) in $R \times B$, an r–b M-path that contains a piercing point.*

Proof. We use the algorithm of Soto and Telha [18] to compute a minimum piercing P of $\mathcal{R}(R, B)$. Then, as we have seen above, $\hat{P}$ is a minimum piercing of $\mathcal{R}(R, B)$, too. Now let (r, b) be a relevant red–blue pair in $R \times B$, and let $\hat{p} \in \hat{P}$ be a piercing point of $R(r, b)$.

Then, by the definition of $A_{\hat{p}}$, the point r lies to the SW of $A_{\hat{p}}$ and b to the NE. We prove that M contains an r–$\hat{p}$ M-path; a symmetric argument proves that M also contains a $\hat{p}$–b M-path. Concatenating these two M-paths yields the desired r–b M-path since b lies to the NE of $\hat{p}$ and $\hat{p}$ lies to the NE of r. Recall that $\hat{p}$ lies on the intersection of the t_W–t_E M-path π_{WE} and the t_S-t_N M-path π_{SN}, where t_W, t_E, t_S, t_N are the terminals that determine the extensions of $A_{\hat{p}}$; see Fig. 2a. To show that M M-connects r and $\hat{p}$, we consider two cases.

Case I: $r \in R(t_W, t_S)$ (dark shaded in Fig. 2b).

According to our assumption, M contains *some* r–b M-path π. It is not hard to see that π must intersect π_{WE} or π_{SN} in some point x to the SW of $\hat{p}$. Thus, we can go, in a monotone fashion, on π from r to x and then on π_{WE} or π_{SN} from x to $\hat{p}$. This is the desired r–$\hat{p}$ M-path.

Case II: r lies to the SW of t_W or t_S; see Fig. 2c.

Clearly, M contains M-paths from r to t_{W} and to t_{S}. If r lies to the SW of t_{W}, we can go, again in a monotone fashion, from r to t_{W} and then on π_{WE} from t_{W} to $\hat{p}$. Otherwise, if r lies to the SW of t_{S}, we can go from r to t_{S} and then on π_{SN} from t_{S} to $\hat{p}$. □

Lemmas 1 and 2 (with $R = T_1$, $B = T_2$, and $M = N_{xy}$) yield the following.

Theorem 1. *We can efficiently compute a 4-approximation for the 2-plane case.*

2.2 Recursive Algorithm for k Planes

Now we extend our 2-plane algorithm to the k-plane case.

Theorem 2. *There is a $4(k-1)$-approximation algorithm for 3D-MMN where the terminals lie in the union of $k \geq 2$ planes parallel to the x–y plane.*

In the remainder of this section, we restrict ourselves to the directional subproblem of M-connecting all relevant terminal pairs. Specifically, we construct a pillar network $N^{\mathrm{ver}}_{\mathrm{dir}}$ of weight at most $(k-1)\|N^{\mathrm{ver}}_{\mathrm{opt}}\|$. As we have argued at the beginning of Section 2, this suffices to prove Theorem 2.

Our pillar-placement algorithm is as follows. Let $i \in \{1, \ldots, k-1\}$. We construct an instance $\mathcal{I}_i$ of BRP where we two-color T_{xy} such that each point corresponding to a terminal of some plane E_j with $j \leq i$ is colored red and each point corresponding to a terminal of some plane $E_{j'}$ with $j' \geq i+1$ is colored blue. For $\mathcal{I}_i$, we compute a minimum piercing $\hat{P}_i$ according to Lemma 2 with $M = N_{xy}$. In other words, for any relevant pair $(t_j, t_{j'})$, there is some M-path in N_{xy} that contains a piercing point of $\hat{P}_i$. We choose $i^\star \in \{1, \ldots, k-1\}$ such that $\hat{P}_{i^\star}$ has minimum cardinality. This is crucial for our analysis. At the piercing points of $\hat{P}_{i^\star}$, we erect pillars spanning all planes $E_1, \ldots, E_k$. Let $\hat{N}_{i^\star}$ be the set of these pillars. We first investigate feasibility of the resulting network.

Lemma 3. *The network $N^{\mathrm{hor}} \cup \hat{N}_{i^\star}$ M-connects any relevant terminal pair $(t_j, t_{j'})$ with $j \leq i^\star$ and $j' \geq i^\star + 1$.*

Proof. Consider a pair $(t_j, t_{j'})$ as in the statement. We construct an M-path from t_j to $t_{j'}$ as follows. We know that there is an M-path π that connects the projections of t_j and $t_{j'}$ in N_{xy} and contains a piercing point p of $\hat{P}_{i^\star}$. So we start in t_j and follow the projection of π onto plane E_j until we arrive in p. Then we use the corresponding pillar in $\hat{N}_{i^\star}$ to reach plane $E_{j'}$ where we follow the projection of π (onto that plane) until we reach $t_{j'}$. □

In order to also M-connect terminals in planes $E_j, E_{j'}$ where either $j, j' \leq i^\star$ or $j, j' \geq i^\star + 1$, we simply apply the pillar-placement algorithm recursively to the sets $\{E_1, \ldots, E_{i^\star}\}$ and $\{E_{i^\star+1}, \ldots, E_k\}$. This yields the desired pillar network $N^{\mathrm{ver}}_{\mathrm{dir}}$. By Lemma 3, $N^{\mathrm{ver}}_{\mathrm{dir}} \cup N^{\mathrm{hor}}$ is feasible. Next, we bound $\|\hat{N}_{i^\star}\|$.

Lemma 4. *Let M be an arbitrary directional Manhattan network for T, and let M^{ver} be the set of vertical segments in M. Then the pillar network $\hat{N}_{i^\star}$ has weight at most $\|M^{\mathrm{ver}}\|$.*

Proof. Without loss of generality, we assume that M is a subnetwork of the Hanan grid [10]. We may also assume that any segment of M^{ver} spans only consecutive planes. For $1 \leq i \leq j \leq k$, let $M_{i,j}$ denote the subnetwork of M^{ver} lying between planes E_i and E_j. Let $d_{i,j}$ be the vertical distance of planes E_i and E_j.

We start with the observation that, for any $j = 1, \ldots, k$, the network $M_{j,j+1}$ is a set of pillars that forms a valid piercing of the piercing instance $\mathcal{I}_j$. Hence, $|M_{j,j+1}| \geq |\hat{P}_j| \geq |\hat{P}_{i^\star}|$, which implies the claim of the lemma as follows:

$$\|M^{\text{ver}}\| = \sum_{j=1}^{k-1} \|M_{j,j+1}\| = \sum_{j=1}^{k-1} |M_{j,j+1}| \cdot d_{j,j+1} \geq \sum_{j=1}^{k-1} |P_{i^\star}| \cdot d_{j,j+1} = \|P_{i^\star}\|.$$

□

It is crucial that the pillars constructed recursively span either $E_1, \ldots, E_{i^\star}$ or $E_{i^\star+1}, \ldots, E_k$ but not all planes. For $1 \leq j \leq j' \leq k$, let $\text{weight}_z(j, j')$ denote the weight of the vertical part of the network produced by the above pillar-placement algorithm when applied to planes $E_j, \ldots, E_{j'}$ recursively. Assume that $j < j'$ and that the algorithm splits at plane $E_{i'}$ with $j \leq i' < j'$ when planes $E_j, \ldots, E_{j'}$ are processed. By means of Lemma 4, we derive the recursion

$$\text{weight}_z(j, j') \leq \|M_{j,j'}\| + \text{weight}_z(j, i') + \text{weight}_z(i' + 1, j') \,,$$

which holds for any M-network M for T. We now claim that

$$\text{weight}_z(j, j') \leq (j' - j)\|M_{j,j'}\|.$$

Our proof is by induction on the number of planes processed by the algorithm. By the inductive hypothesis, we have that $\text{weight}_z(j, i') \leq (i' - j)\|M_{j,i'}\|$ and $\text{weight}_z(i' + 1, j') \leq (j' - i' - 1)\|M_{i'+1,j'}\|$. Since $\|M_{j,i'}\| + \|M_{i'+1,j'}\| \leq \|M_{j,j'}\|$ and $\text{weight}_z(l, l) = 0$ for any $l \in \{1, \ldots, k\}$, the claim follows.

We conclude that the weight of the solution produced by the algorithm when applied to all planes $E_1, \ldots, E_k$ is bounded by $\text{weight}_z(1, k) \leq (k - 1)\|M_{1,k}\| = (k - 1)\|M^{\text{ver}}\|$. This finishes the proof of Theorem 2.

3 General Case

In this section, we present an approximation algorithm, the *grid algorithm*, for general 3D-MMN. Our main result is the following.

Theorem 3. *For any $\varepsilon > 0$, there is an $O(n^\varepsilon)$-approximation algorithm for 3D-MMN that runs in time $n^{O(1/\varepsilon)}$.*

Our approach extends to higher dimensions; see the full version of our paper [7].

For technical reasons we assume that the terminals are in general position, that is, any two terminals differ in all three coordinates. As in the k-plane case it suffices to describe and analyze the algorithm for the directional subproblem.

Algorithm. We start with a high-level summary. To solve the directional subproblem, we place a 3D-grid that partitions the instance into a constant number of cuboids. Cuboids that differ in only two coordinates form *slabs*. We connect terminals from different slabs by M-connecting each terminal to the corners of its cuboid and by using the edges of the grid to connect the corners. We connect terminals from the same slab by recursively applying our algorithm to the slabs.

Step 1: Partitioning into cuboids and slabs. Consider the bounding cuboid C of T and set $c = 3^{1/\varepsilon}$. Partition C by $3(c-1)$ separating planes into $c \times c \times c$ axis-aligned subcuboids C_{ijk} with $i, j, k \in \{1, \ldots, c\}$. The indices are such that larger indices mean larger coordinates. Place the separating planes such that the number of terminals between two consecutive planes is at most n/c. This can be accomplished by a simple plane-sweep for each direction x, y, z, and placing separating planes after n/c terminals. Here we exploit our general-position assumption. The edges of the resulting subcuboids—except the edges on the boundary of C which we do not need—induce a three-dimensional grid $\mathcal{G}$ of axis-aligned line segments. We insert $\mathcal{G}$ into the solution.

For each $i \in \{1, \ldots, c\}$, define the x-aligned *slab*, C_i^x, to be the union of all cuboids C_{ijk} with $j, k \in \{1, \ldots, c\}$. Define y-aligned and z-aligned slabs C_j^y, C_k^z analogously; see the full version of this paper [7] for figures.

Step 2: Add M-paths between different slabs. Consider two cuboids C_{ijk} and $C_{i'j'k'}$ with $i < i'$, $j < j'$, and $k < k'$. Any terminal pair $(t, t') \in C_{ijk} \times C_{i'j'k'}$ can be M-connected using the edges of $\mathcal{G}$ as long as t and t' are connected to the appropriate corners of their cuboids; see the full version [7] for a figure. To this end, we use the following *patching* procedure.

Call a cuboid C_{ijk} *relevant* if there is a non-empty cuboid $C_{i'j'k'}$ with $i < i'$, $j < j'$, and $k < k'$. For each relevant cuboid C_{ijk}, let $\hat{p}_{ijk}$ denote a corner that is dominated by all terminals inside C_{ijk}. By *Up-patching* C_{ijk} we mean to M-connect every terminal in C_{ijk} to $\hat{p}_{ijk}$. We up-patch C_{ijk} by solving (approximately) an instance of 3D-RSA with the terminals in C_{ijk} as points and $\hat{p}_{ijk}$ as origin. We define *down-patching* symmetrically; cuboid C_{ijk} is relevant if there is a non-empty cuboid $C_{i'j'k'}$ with $i > i'$, $j > j'$, $k > k'$ and $\check{p}_{ijk}$ is the corner that dominates all terminals in C_{ijk}.

We insert the up- and down-patches of all relevant cuboids into the solution.

Step 3: Add M-paths within slabs. To M-connect relevant terminal pairs that lie in the same slab, we apply the grid algorithm (steps 1–3) recursively to each slab C_i^x, C_j^y, and C_k^z with $i, j, k \in \{1, \ldots, c\}$.

Analysis. We first show that the output of the grid algorithm is feasible, then we establish its approximation ratio of $O(n^\varepsilon)$ and its running time of $n^{O(1/\varepsilon)}$ for any $\varepsilon > 0$. In this section, OPT denotes the weight of a minimum M-network (*not* the cost of an optimal solution to the directional subproblem).

Lemma 5. *The grid algorithm M-connects all relevant terminal pairs.*

Proof. Let (t, t') be a relevant terminal pair.

First, suppose that t and t' lie in cuboids of different slabs. Thus, there are $i < i', j < j', k < k'$ such that $t \in C_{ijk}$ and $t' \in C_{i'j'k'}$. Furthermore, C_{ijk} and $C_{i'j'k'}$ are relevant for up- and down-patching, respectively. When up-patching, we solve an instance of RSA connecting all terminals in C_{ijk} to $\hat{p}_{ijk}$. Similarly, down-patching M-connects t' to $\check{p}_{i'j'k'}$. The claim follows as $\mathcal{G}$ M-connects $\hat{p}_{ijk}$ and $\check{p}_{i'j'k'}$.

Now, suppose that t and t' lie in the same slab. As the algorithm is applied recursively to each slab, there is a recursion step where t and t' lie in cuboids in different slabs. Here, we need our general-position assumption. Applying the argument above to that particular recursive step completes the proof. □

Next, we turn to the performance of our algorithm. Let $r(n)$ be its approximation ratio, where n is the number of terminals in T. The total weight of the output is the sum of $\|\mathcal{G}\|$, the cost of patching, and the cost for the recursive treatment of the slabs. We analyze each of the three costs separately.

The grid $\mathcal{G}$ consists of all edges induced by the c^3 subcuboids except the edges on the boundary of C. Let ℓ denote the length of the longest side of C. The weight of $\mathcal{G}$ is at most $3(c-1)^2\ell$, which is bounded by $3c^2\mathrm{OPT}$ as $\mathrm{OPT} \geq \ell$.

Let $r_{\mathrm{patch}}(n)$ denote the cost of patching all relevant cuboids in step 2. Lemma 6 (given below) proves that $r_{\mathrm{patch}}(n) = O(n^{\varepsilon})\mathrm{OPT}$.

Now consider the recursive application of the algorithm to all slabs. Recall that N_{opt} is a fixed minimum M-network for T. For $i \in 1, \ldots, c$, let OPT_i^x be the optimum cost for M-connecting *all* (not only relevant) terminal pairs in slab C_i^x. Costs OPT_i^y and OPT_i^z are defined analogously.

Slightly abusing of notation, we write $N_{\mathrm{opt}} \cap C_i^x$ for the set of line segments of N_{opt} that are completely contained in slab C_i^x. Observe that $N_{\mathrm{opt}} \cap C_i^x$ forms a feasible solution for C_i^x. Thus, $\mathrm{OPT}_i^x \leq \|N_{\mathrm{opt}} \cap C_i^x\|$. By construction, any slab contains at most n/c terminals. Hence, the total cost of the solutions for slabs $C_1^x, \ldots, C_c^x$ is at most

$$\sum_{i=1}^{c} r\left(\frac{n}{c}\right) \mathrm{OPT}_i^x \leq r\left(\frac{n}{c}\right) \sum_{i=1}^{c} \|N_{\mathrm{opt}} \cap C_i^x\| \leq r\left(\frac{n}{c}\right) \mathrm{OPT}\,.$$

Clearly, the solutions for the y- and z-slabs have the same bound.

Summing up all three types of costs, we obtain the recursive equation

$$r(n)\mathrm{OPT} \leq 3c^2\mathrm{OPT} + 3r\left(\frac{n}{c}\right) \mathrm{OPT} + r_{\mathrm{patch}}(n)\mathrm{OPT}\,.$$

Hence, $r(n) = O(n^{\max\{\varepsilon, \log_c 3\}})$. Plugging in $c = 3^{1/\varepsilon}$ yields $r(n) = O(n^{\varepsilon})$, which proves the approximation ratio claimed in Theorem 3.

Lemma 6. *Patching all relevant cuboids costs* $r_{\mathrm{patch}}(n) \in O(n^{\varepsilon})\mathrm{OPT}$.

Proof. It suffices to consider up-patching; down-patching is symmetric.

Lemma 7 shows that by reducing the patching problem to 3D-RSA, we can find such a network of cost $O(\rho)\mathrm{OPT}$, where ρ is the approximation factor of 3D-RSA.

In the full version of our paper [7], we argue that there is an approximation preserving reduction from 3D-RSA to DST. DST, in turn, admits an $O(n^\varepsilon)$-approximation for any $\varepsilon > 0$ [4]. Hence, the cost of up-patching is indeed bounded by $O(n^\varepsilon)$OPT. □

The following lemma is proved in the full version [7] of this paper.

Lemma 7. *Given an efficient ρ-approximation of 3D-RSA, we can efficiently up-patch all relevant cuboids at cost no more than $12(c^2 + c)\rho$OPT.*

Finally, we analyze the running time. Let $T(n)$ denote the running time of the algorithm applied to a set of n terminals. The running time is dominated by patching and the recursive slab treatment. Using the DST algorithm of Charikar et al. [4], patching cuboid C_i requires time $n_i^{O(1/\varepsilon)}$, where n_i is the number of terminals in C_i. As each cuboid is patched at most twice and there are c^3 cuboids, patching takes $O(c^3)n^{O(1/\varepsilon)} = n^{O(1/\varepsilon)}$ time. The algorithm is applied recursively to $3c$ slabs. This yields the recurrence $T(n) = 3cT(n/c) + n^{O(1/\varepsilon)}$, which leads to the claimed running time.

4 Open Problems

We have presented a grid-based $O(n^\varepsilon)$-approximation for dD-MMN. This is a significant improvement over the ratio of $O(n^{4/5+\varepsilon})$ which is achieved by reducing the problem to DSF [8]. For 3D, we have described a $4(k-1)$-approximation for the case when the terminals lie on $k \geq 2$ parallel planes. This outperforms our grid-based algorithm when $k \in o(n^\varepsilon)$. Whereas 2D-MMN admits a 2-approximation [5,11,16], it remains open whether $O(1)$- or $O(\log n)$-approximations can be found in higher dimensions.

Our $O(n^\varepsilon)$-approximation algorithm for dD-MMN solves instances of dD-RSA for the subproblem of patching. We conjecture that dD-RSA admits a PTAS, which is known for 2D-RSA [14]. While this is an interesting open question, a positive result would still not be enough to improve our approximation ratio, which is dominated by the cost of finding M-paths inside slabs.

The complexity of the *undirectional* bichromatic rectangle piercing problem (see Section 2) is still unknown. Currently, the best approximation has a ratio of 4, which is (trivially) implied by the result of Soto and Telha [18]. Any progress would immediately improve the approximation ratio of our algorithm for the k-plane case of 3D-MMN (for any $k > 2$).

Acknowledgments. This work was started at the 2009 Bertinoro Workshop on Graph Drawing. We thank the organizers Beppe Liotta and Walter Didimo for creating an inspiring atmosphere. We also thank Steve Wismath, Henk Meijer, Jan Kratochvíl, and Pankaj Agarwal for discussions. We are indebted to Stefan Felsner for pointing us to Soto and Telha's work [18].

The research of Aparna Das and Stephen Kobourov was funded in part by NSF grants CCF-0545743 and CCF-1115971.

References

1. Arora, S.: Approximation schemes for NP-hard geometric optimization problems: A survey. Math. Program. 97(1–2), 43–69 (2003),
2. Arya, S., Das, G., Mount, D.M., Salowe, J.S., Smid, M.: Euclidean spanners: Short, thin, and lanky. In: Proc. 27th Annu. ACM Symp. Theory Comput. (STOC), pp. 489–498. ACM Press, New York (1995)
3. Benkert, M., Wolff, A., Widmann, F., Shirabe, T.: The minimum Manhattan network problem: Approximations and exact solutions. Comput. Geom. Theory Appl. 35(3), 188–208 (2006)
4. Charikar, M., Chekuri, C., Cheung, T.Y., Dai, Z., Goel, A., Guha, S., Li, M.: Approximation algorithms for directed Steiner problems. In: Proc. 9th ACM-SIAM Sympos. Discrete Algorithms (SODA), pp. 192–200 (1998)
5. Chepoi, V., Nouioua, K., Vaxès, Y.: A rounding algorithm for approximating minimum Manhattan networks. Theor. Comput. Sci. 390(1), 56–69 (2008)
6. Chin, F., Guo, Z., Sun, H.: Minimum Manhattan network is NP-complete. Discrete Comput. Geom. 45, 701–722 (2011),
7. Das, A., Gansner, E.R., Kaufmann, M., Kobourov, S., Spoerhase, J., Wolff, A.: Approximating minimum Manhattan networks in higher dimensions. ArXiv e-print abs/1107.0901 (2011)
8. Feldman, M., Kortsarz, G., Nutov, Z.: Improved approximating algorithms for directed Steiner forest. In: Proc. 20th ACM-SIAM Sympos. Discrete Algorithms (SODA), pp. 922–931 (2009)
9. Fuchs, B., Schulze, A.: A simple 3-approximation of minimum Manhattan networks. Tech. Rep. 570, Zentrum für Angewandte Informatik Köln (2008)
10. Gudmundsson, J., Levcopoulos, C., Narasimhan, G.: Approximating a minimum Manhattan network. Nordic J. Comput. 8, 219–232 (2001)
11. Guo, Z., Sun, H., Zhu, H.: Greedy construction of 2-approximation minimum Manhattan network. In: Hong, S., Nagamochi, H., Fukunaga, T. (eds.) ISAAC 2008. LNCS, vol. 5369, pp. 4–15. Springer, Heidelberg (2008)
12. Kato, R., Imai, K., Asano, T.: An improved algorithm for the minimum Manhattan network problem. In: Bose, P., Morin, P. (eds.) ISAAC 2002. LNCS, vol. 2518, pp. 344–356. Springer, Heidelberg (2002),
13. Lam, F., Alexandersson, M., Pachter, L.: Picking alignments from (Steiner) trees. J. Comput. Biol. 10, 509–520 (2003)
14. Lu, B., Ruan, L.: Polynomial time approximation scheme for the rectilinear Steiner arborescence problem. J. Comb. Optim. 4(3), 357–363 (2000)
15. Muñoz, X., Seibert, S., Unger, W.: The minimal Manhattan network problem in three dimensions. In: Das, S., Uehara, R. (eds.) WALCOM 2009. LNCS, vol. 5431, pp. 369–380. Springer, Heidelberg (2009)
16. Nouioua, K.: Enveloppes de Pareto et Réseaux de Manhattan: Caractérisations et Algorithmes. Ph.D. thesis, Université de la Méditerranée (2005)
17. Seibert, S., Unger, W.: A 1.5-approximation of the minimal Manhattan network problem. In: Deng, X., Du, D. (eds.) ISAAC 2005. LNCS, vol. 3827, pp. 246–255. Springer, Heidelberg (2005)
18. Soto, J.A., Telha, C.: Jump number of two-directional orthogonal ray graphs. In: Gülük, O., Woeginger, G. (eds.) IPCO 2011. LNCS, vol. 6655, pp. 389–403. Springer, Heidelberg (2011)
19. Zelikovsky, A.: A series of approximation algorithms for the acyclic directed Steiner tree problem. Algorithmica 18(1), 99–110 (1997)