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 study 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 algorithm. For 3D, we also give a $4(k-1)$-approximation algorithm for the case that the terminals are contained in the union of $k \geq 2$ parallel planes.

Keywords: Approximation Algorithms, Computational Geometry, Minimum Manhattan Network

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 of little cost. 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 low-cost 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. Rather than requiring that the total length of the network is proportional to the minimum spanning tree of the points, 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 between 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, 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*
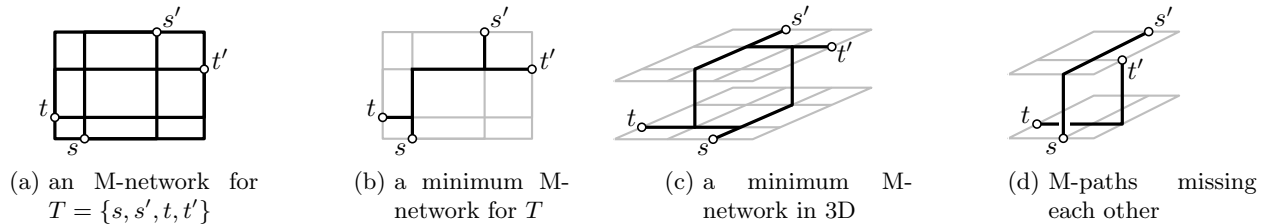
Fig. 1: Examples of M-networks in 2D and 3D.

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 several areas such as VLSI layout and computational biology. For example, Lam et al. [12] used them in gene alignment in order to reduce the size of the search space of the Viterbi algorithm for pair hidden Markov models.

1.1 Previous work

The 2D-version of the problem, 2D-MMN, was introduced by Gudmundsson et al. [9]. They gave an 8- and a 4-approximation algorithm. Later, the approximation ratio was improved to 3 [3,8] and then to 2, which is currently the best possible. It was achieved in three different ways: via linear programming [5], using the primal–dual scheme [15] and with purely geometric arguments [10]. The last two algorithms run in $O(n \log n)$ time, given a set of n points in the plane. A ratio of 1.5 was claimed [16], but apparently the proof is incomplete [8]. Chin et al. [6] finally settled the complexity of 2D-MMN by proving it NP-hard.

A little earlier, Muñoz et al. [14] considered 3D-MMN. They showed that the problem is NP-hard and that it is 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 algorithm, where α denotes the best approximation ratio for 2D-MMN. They posed the design of approximation algorithms for general 3D-MMN as an open problem.

1.2 Related problems

As we observe in Section 2.3, 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. [7] reported, for any $\varepsilon > 0$, an $O(n^{4/5+\varepsilon})$-approximation algorithm for DSF, where n is the number of vertices of the given graph. 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 an edge-weighted digraph G, a *root* vertex r, and a subset S of the vertices of G to which r must connect. An optimum solution for DST is a minimum-weight 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 in MMN, the

aim is to find a minimum-weight network. For 2D-RSA, there is a polynomial-time approximation scheme (PTAS) [13] based on Arora's technique for approximating geometric optimization problems such as TSP [1]. It is not known whether 2D-MMN admits a PTAS. Arora's technique does not directly apply here as M-paths between terminals forbid detours and thus may not respect portals.

1.3 Our contribution

We first 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 3.

Our main result is an $O(n^\varepsilon)$-approximation algorithm for dD-MMN, for any $\varepsilon > 0$. We first present the algorithm in detail for three dimensions; see Section 4. Since the algorithm for arbitrary dimensions is a straightforward generalization of the algorithm for 3D but less intuitive, we describe it in the appendix.

Our $O(n^\varepsilon)$-approximation algorithm for dD-MMN constitutes a significant improvement upon the best known ratio of $O(n^{4/5+\varepsilon})$ for (general) directed Steiner forest [7]. 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 [18].

Our $O(k)$-approximation algorithm for the k-planes case relies on recent work by Soto and Telha [17]. They show 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. Combining this result with an approximation algorithm for 2D-MMN, yields an approximation algorithm for the 2-planes case. We show how to generalize this idea to k planes.

2 Some Basic Observations

We begin with some notation. 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*. In what follows, we consider the 3-dimensional case of the MMN problem, unless otherwise stated.

2.1 Quadratic Lower Bound for Generating Sets in 3D

Intuitively, what makes 3D-MMN more difficult than 2D-MMN is the following: 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* [11] 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 [11]. Unfortunately this result does not extend to 3D. Below, we construct an instance that requires a generating set of size $\Omega(n^2)$. The idea of using linear-size generating sets is exploited by several algorithms for 2D-MMN [5,11]. The following theorem shows that these approaches do not easily carry over to 3D.

3

Theorem 1. *There exists an instance of 3D-MMN with n terminals that requires a generating set of size $\Omega(n^2)$.*

Proof. We construct an instance that requires a generating set of size at least $n^2/4$. The main idea of the construction is to ensure that $n^2/4$ of the terminal pairs must use an edge segment unique to that specific pair. The input consists of two sets T and T', each with $n/2$ terminals, with the following coordinates: for $0 \le i < n/2$, terminal $t_i \in T$ is at $(i, n/2 - i, n/2 - i)$ and terminal $t'_i \in T'$ is at $(n/2 + i, n - i, n - i)$. Figure 2 shows the instance for $n = 6$.
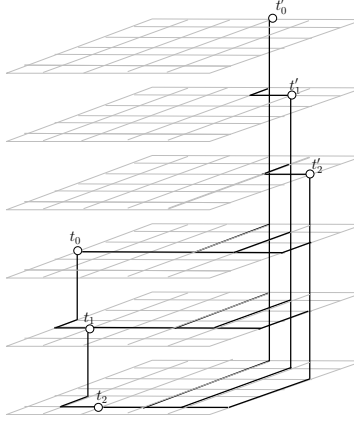

Fig. 2: The constructed network for $n = 6$.

Consider any given generating set $Z \subset T \times T'$ such that there is a pair $(\tilde{t}, \tilde{t}')$, $\tilde{t} \in T$ and $\tilde{t}' \in T'$ that is not in Z. We now construct a specific network that contains M-paths between all terminal pairs in Z but no M-path between $(\tilde{t}, \tilde{t}')$.

Consider any pair $(t, t') \in Z$ such that $t = (i, j, k) \in T$ and $t' = (i', j', k') \in T'$. The M-path from t to t' has three segments: an x-aligned segment from (i, j, k) to (i', j, k), a y-aligned segment from (i', j, k) to (i', j', k), and a z-aligned segment from (i', j', k) to (i', j', k'). To ensure an M-path between each generating pair $t_i, t_j \in T$ (similarly between $t'_i, t'_j \in T'$), we add M-paths between each pair of consecutive terminals in T (similarly for T') as follows: we connect $t_i, t_{i+1} \in T$ by adding a z-aligned segment from $t_i = (i, j, k)$ to $(i, j, k-1)$, a y-aligned segment to $(i, j-1, k-1)$, and an x-aligned segment to $t_{i+1} = (i+1, j-1, k-1)$; see Fig. 2.

It is easy to verify that, in this construction, the M-path between terminals $t = (i, j, k) \in T$ and $t' = (i', j', k') \in T'$ must use the y-aligned segment between (i', j, k) and (i', j', k). Since this segment is added only between terminal pairs that are present in the generating set Z, there is no M-path between terminals $\tilde{t} \in T$ and $\tilde{t}' \in T'$ which are not in Z. Thus, in order to obtain M-paths between all pairs of terminals in $T \cup T'$, we need at least all of the $n^2/4$ pairs in $T \times T'$. □

2.2 Hanan Grid and Directional Subproblems

First, we note that any instance of MMN has a solution that is contained in the *Hanan grid*, the grid induced by the terminals; see Fig. 1(a). Gudmundsson et al. [9] showed this for 2D; their proof generalizes to higher dimensions. In what follows, we restrict ourselves to finding feasible solutions that are contained in the Hanan grid.

4

Second, to simplify our proofs, we consider the *directional* subproblem of 3D-MMN which consists of 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*.

The idea behind our reduction to the directional subproblem is that any instance of 3D-MMN can be decomposed into four subproblems of this type. One may think of the above-defined directional subproblem as connecting the terminals which are oriented in a north-east (NE) configuration in the x–y plane (with increasing z-coordinates). Analogous subproblems exist for the directions NW, SE, and SW. Note that any terminal pair belongs to one of these four categories (if seen from the terminal with smaller z-coordinate).

The decomposition extends to higher dimensions d, by fixing the relationship between (t, t') for one dimension (for example, z), and enumerating over all possible relationships for the remaining $d - 1$ dimensions. This decomposes dD-MMN into 2^{d-1} subproblems, which is a constant number of subproblems as we consider d to be a fixed constant.

This means that any ρ-approximation algorithm for the directional subproblem leads to an $O(\rho)$-approximation algorithm for the general case. Thus we can focus on designing algorithms for the directional subproblem.

Observation 1 *Any instance of MMN can be decomposed into a constant number of directional subproblems. Thus a ρ-approximation algorithm for the directional subproblem leads to an $O(\rho)$-approximation algorithm for MMN.*

2.3 Relation to Steiner Problems

We next show that there is an approximation-preserving reduction from directional 3D-MMN to the directed Steiner forest (DSF) problem, which by Observation 1, carries over up to a constant factor, to general 3D-MMN.

Let T be a set of n points in $\mathbb{R}^3$. Let H be the Hanan grid induced by T. We consider H as an undirected graph where the length of each edge equals the Euclidean distance between its endpoints. We orient each edge in H so that, for any edge (p, p') in the resulting digraph H', the start node p dominates the end node p'. We call H' the *oriented Hanan grid* of T. Now let (t, t') be a relevant pair of points in T, that is, t dominates t'. Any M-path in H connecting t to t' corresponds to a directed path in H' from t to t'. The converse also holds: every directed path in H' corresponds to an M-path in H.

Let I be an instance of directional 3D-MMN and let I' be an instance of DSF where the input graph is H' and where every relevant terminal pair of I has to be connected. Then, each feasible solution N of I contained in H corresponds to a sub-graph N' of H' that connects every relevant terminal pair, and is therefore a feasible solution to I'. It is easy to see that N' has the same cost as N, as N' uses the oriented version of each edge of N. Conversely, every feasible solution N' for I' corresponds to a subgraph N of H that M-connects every relevant terminal pair. Therefore, N is a feasible solution to I' with the same cost as N'. This establishes an efficiently computable one-to-one correspondence between feasible solutions to I that are contained in H and feasible solutions to I'. Since there is an optimum solution to I contained in H [9], this is an approximation-preserving reduction from directional 3D-MMN to DSF.

By means of the above transformation of the Hanan grid into a digraph, we also obtain an approximation-preserving reduction from 3D-RSA to DST. We use this later in Section 4 to develop an approximation algorithm for 3D-MMN. Let I be an instance of 3D-RSA given by a set T of

terminals with non-negative coordinates that are to be M-connected to the origin o. We construct an instance I' of DSF as above where $\{o\} \times T$ is the set of node pairs to be connected. Note that any feasible solution to I' is, without loss of generality, a tree. Hence, I' is an instance of DST with root o. All in all, we have an approximation-preserving reduction from 3D-RSA to DST.

3   The 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 4 if $k \in o(n^\varepsilon)$.

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

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 (that is, z-aligned) line segments. Finally, the algorithm returns the set $N = N^{\mathrm{hor}} \cup N^{\mathrm{ver}}$.

Phase I is simple; we compute a 2-approximate M-network N_{xy} for T_{xy} (using the algorithm of Guo et al. [10]) 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 2 $\|N^{\mathrm{hor}}\| \leq 2k\|N_{\mathrm{opt}}^{\mathrm{hor}}\|$.

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

In Phase II, we construct a *pillar network* by computing a set N^{ver} of vertical line segments, so-called *pillars*, of total cost at most $4(k-1)\|N_{\mathrm{opt}}^{\mathrm{ver}}\|$. This yields an overall approximation factor of $4(k-1)$ since $\|N^{\mathrm{hor}} \cup N^{\mathrm{ver}}\| \leq 2k\|N_{\mathrm{opt}}^{\mathrm{hor}}\| + 4(k-1)\|N_{\mathrm{opt}}^{\mathrm{ver}}\| \leq 4(k-1)(\|N_{\mathrm{opt}}^{\mathrm{hor}}\| + \|N_{\mathrm{opt}}^{\mathrm{ver}}\|) \leq 4(k-1)\mathrm{OPT}$.

Below we describe Phase II of our algorithm for the directional subproblem that runs in direction north-east (NE) in the x–y plane (with increasing z-coordinates). For this directional subproblem, we construct a pillar network $N_{\mathrm{dir}}^{\mathrm{ver}}$ of weight at most $(k-1)\|N_{\mathrm{opt}}^{\mathrm{ver}}\|$ that, together with N^{hor}, M-connects all relevant pairs. We solve the analogous subproblems for the directions NW, SE, and SW in the same fashion. Then N^{ver} is the union of the four partial solutions and has weight at most $4(k-1)\|N_{\mathrm{opt}}^{\mathrm{ver}}\|$, 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 [17] 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.

The details of Phase II appear, for $k = 2$ planes, in Section 3.1, and, for $k > 2$ planes, in Section 3.2. Algorithm 1 summarizes of our k-planes algorithm.

Algorithm 1 k-Planes Algorithm
___
Input: Set T of terminals contained in the union of planes $E_1, \ldots, E_k$, all parallel to the x–y plane.

1: Let T_{xy} be the projection of T onto the x–y plane
2: Phase I:
    Compute N_{xy}, a 2-approximate M-network for T_{xy} using the algorithm of Guo et al. [10].
    Let N^{hor} be the union of the projections of N_{xy} onto each of the planes $E_1, \ldots, E_k$.
3: Phase II:
    If $k = 2$, construct a pillar network N^{ver} by Algorithm 2; see Section 3.1.
    Otherwise, construct a pillar network N^{ver} by Algorithm 3; see Section 3.2.
Output: $N^{\text{hor}} \cup N^{\text{ver}}$.

3.1 Pillar Network for Two Planes

Our phase-II algorithm for two planes is very simple. We sketch it first in order to provide some intuition for the k-planes case. Let the terminals in T_1 be red and those in T_2 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.

Algorithm 2 Pillar network of the directional subproblem for $k = 2$ planes
___
Input: Sets $T_1 \subset E_1$ and $T_2 \subset E_2$ of terminals.

1: Color T_1 red and T_2 blue.
2: Ignoring z-coordinates of terminals, let $\mathcal{R}(T_1, T_2)$ be the set of rectangles spanned by relevant red–blue pairs.
3: Compute a minimum piercing $\hat{P}$ of $\mathcal{R}(T_1, T_2)$ such that for each relevant red–blue pair $(r, b) \in T_1 \times T_2$ the piercing point for (r, b) lies on an r–b M-path in N_{xy}, as described in Lemma 2.
4: Erect pillars from E_1 to E_2 at each piercing point $\hat{p} \in \hat{P}$; let $N_{\text{dir}}^{\text{ver}}$ be the resulting set of pillars.
Output: $N_{\text{dir}}^{\text{ver}}$.

Our algorithm (Algorithm 2) 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 [17]. 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 $N_{\text{dir}}^{\text{ver}}$ be the corresponding set of pillars.

Lemma 1. *It holds that* $\|N_{\text{dir}}^{\text{ver}}\| \leq \|N_{\text{opt}}^{\text{ver}}\|$.

Proof. It is easy to see that $|\hat{P}| = |P|$. Integrating over the distance d of the two planes yields $\|N_{\text{dir}}^{\text{ver}}\| = |\hat{P}| \cdot d = |P| \cdot d \leq \|N_{\text{opt}}^{\text{ver}}\|$. 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 $N_{\text{opt}}^{\text{ver}}$ pierce $\mathcal{R}(T_1, T_2)$—otherwise N_{opt} would not be feasible. $\qquad\square$

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}.

For a piercing point $p \in P$, let A_p be the intersection of the relevant rectangles pierced by p; see Fig. 3. Clearly, $p \in A_p$. Note that the bottom and left sides of A_p are determined by terminals t_W and t_S to the west and south 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. Terminals t_W and t_S may coincide, and so may t_E and t_N. It is easy to see that 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.
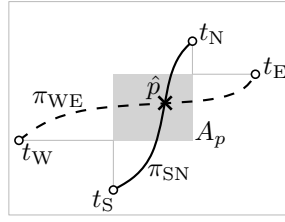

Fig. 3: Paths π_{SN} and π_{WE} meet in a point $\hat{p}$ in A_p.

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 [17] 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 $p \in P$ be a point that pierces $R(r, b)$. Clearly, $\hat{p}$ pierces $R(r, b)$, too. As we have observed before, both p and $\hat{p}$ lie in A_p.

Since (r, b) is a relevant pair, r lies to the SW of A_p and b to the NE; see Fig. 4a. 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 r lies to the SW of $\hat{p}$ and $\hat{p}$ lies to the SW of b. 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_p; see Fig. 3. To show that M M-connects r and $\hat{p}$, we consider two cases.

Case I: $r \in R(t_W, t_S)$; see Fig. 4a. According to our assumption, M contains *some* r–b M-path π. Then π must intersect π_{WE} or π_{SN} at some point x to the SW of $\hat{p}$. Thus, we can go, in a monotone fashion, along π from r to x and then along 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. 4b. In this case 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

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

(b) Case II: we can go from r via t_{W} or t_{S} to $\hat{p}$.

Fig. 4: Sketches for the proof of Lemma 2.

then along 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 $\hat{p}$.

Since these are the only two possibilities, this concludes the proof. $\qquad\square$

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

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

3.2 Pillar Network for k Planes

Now we show how our phase-II algorithm generalizes to k planes. As in the 2-planes case, we restrict ourselves to the directional subproblem and construct a pillar network $N_{\mathrm{dir}}^{\mathrm{ver}}$ of weight at most $(k-1)\|N_{\mathrm{opt}}^{\mathrm{ver}}\|$. As we have argued at the beginning of Section 3, this suffices to prove Theorem 3.

Theorem 3. *There exists 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.*

Our pillar-placement algorithm (Algorithm 3) 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 now show that $\hat{N}_{i^\star}$, along with N^{hor}, creates a feasible network for any relevant terminal pair (t_j, t'_j) such that $j \leq i^\star$ and $j' \geq i^\star + 1$.

Lemma 3. *The network $N^{\mathrm{hor}} \cup \hat{N}_{i^\star}$ M-connects any relevant terminal pair in $T_j \times T_{j'}$ with $j \leq i^\star$ and $j' \geq i^\star + 1$.*

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

Algorithm 3 Pillar network for the directional subproblem for $k > 2$ planes
---
Input: Sets $T_s \subset E_s, \ldots, T_t \subset E_t$ of terminals with $s \leq t$ (initially $s = 1$ and $t = k$).

1: Let T' be the projection of $T_s \cup \cdots \cup T_t$ onto the x–y plane.
2: **for each** $i \in \{s, \ldots, t\}$ **do**
3:     Let $\mathcal{I}_i$ be an instance of BRP where each point in T', corresponding to a terminal in T_j with $j \leq i$, is colored red and each point in T', corresponding to a terminal in $T_{j'}$ with $j' \geq i + 1$, is colored blue.
4:     Compute a minimum piercing $\hat{P}_i$ according to Lemma 2 with $M = N_{xy}$.
5: **end for**
6: Choose $i^\star \in \{s, \ldots, t\}$ such that $\hat{P}_{i^\star}$ has minimum cardinality.
7: Let $\hat{N}_{i^\star}$ be the set of pillars erected at each piercing point of $\hat{P}_{i^\star}$, spanning planes $E_s, \ldots, E_t$.
8: Let $\hat{N}_{\leq i^\star}$ be the output of this algorithm applied recursively to $T_s, \ldots, T_{i^\star}$.
9: Let $\hat{N}_{>i^\star}$ be the output of this algorithm applied recursively to $T_{i^\star+1}, \ldots, T_t$.
Output: $\hat{N}_{i^\star} \cup \hat{N}_{\leq i^\star} \cup \hat{N}_{>i^\star}$

In order to also M-connect relevant terminal pairs in $T_j \times T_{j'}$, where either $(j \leq i^\star$ and $j' \leq i^\star)$ or $(j \geq i^\star + 1$ and $j' \geq i^\star + 1)$, we simply apply the pillar-placement algorithm recursively to the sets $T_1, \ldots, T_{i^\star}$ and $T_{i^\star+1}, \ldots, T_k$. This yields the desired pillar network $N_{\mathrm{dir}}^{\mathrm{ver}}$. By Lemma 3, $N_{\mathrm{dir}}^{\mathrm{ver}} \cup N^{\mathrm{hor}}$ 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 [9]. 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 between planes E_i and E_j.

We start with the observation that, for any $j = 1, \ldots, k - 1$, the network $M_{j,j+1}$ is a set of pillars that forms a valid piercing of the piercing instance $\mathcal{I}_j$ (defined right after Theorem 3). Hence, $|M_{j,j+1}| \geq |\hat{P}_j| \geq |\hat{P}_{i^\star}|$, which implies the claim of the lemma as follows:

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

$\square$

It is crucial for our construction 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 $\mathrm{weight}_z(j, j')$ 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. For technical reasons we set $\mathrm{weight}_z(j, j) = 0$. Now assume that $j < j'$ and that the algorithm makes the partition 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

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

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

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

10

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 $\text{weight}_z(i' + 1, j') \leq (j' - i' - 1)\|M_{i'+1,j'}\|$. We plug these expressions into the recursion 1. 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 completes the proof of Theorem 3.

4 The General Case

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

Theorem 4. *For any $\varepsilon > 0$, there exists an $O(n^\varepsilon)$-approximation algorithm for 3D-MMN that, given a set of n terminals, runs in $n^{O(1/\varepsilon)}$ time.*

This result is better than the one in the previous section if the given set of terminals is distributed over $\omega(n^\varepsilon)$ horizontal planes. Moreover, the approach in this section extends to higher dimensions; see appendix.

For technical reasons, we assume that the terminals are in general position, that is, any two terminals differ in all three coordinates. By Observation 1 it suffices to describe and analyze the algorithm for the directional subproblem.

4.1 The 3D Grid Algorithm

We begin the description 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; see Fig. 5a. 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 executing a simple plane-sweep for each direction x, y, z, and by placing separating planes after every 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 Fig. 5b.

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 Fig. 5c. 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}.

(a) Example of a grid with two cuboids from different slabs.

(b) Examples of x-, y-, and z-aligned slabs.

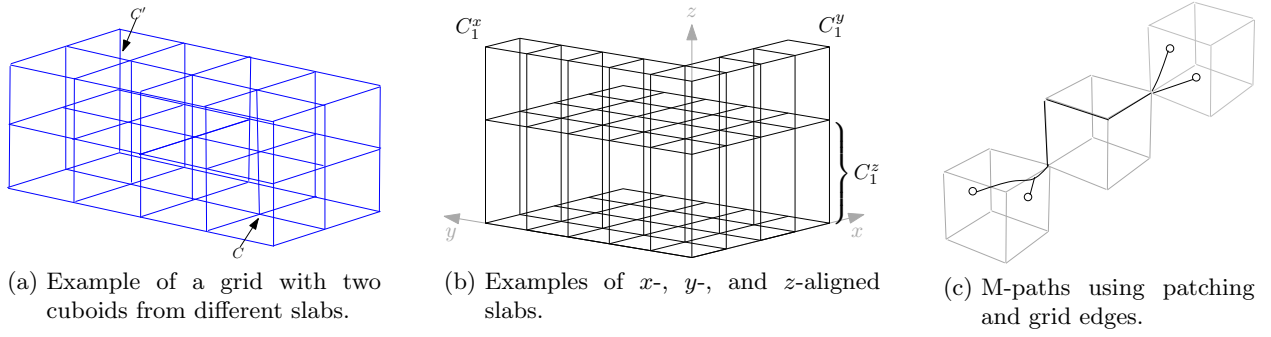(c) M-paths using patching and grid edges.

Fig. 5: Illustrations for the grid algorithm: cuboids, slabs, and patches.

We define *up-patching* C_{ijk} to mean M-connecting 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* analogously; cuboid C_{ijk} is relevant if there is a non-empty cuboid $C_{i'j'k'}$ with $i > i'$, $j > j'$, $k > k'$; we let $\check{p}_{ijk}$ be the corner that dominates all terminals in C_{ijk}.

We complete this step by inserting the up-patches and the 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 $i, j, k \in \{1, \dots, c\}$.

4.2 Analysis

We first show that the output of the algorithm presented in Section 4.1 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 (Feasibility). *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 will be 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. □

Approximation ratio. 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 $\ell \leq \mathrm{OPT}$.

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. Define OPT_i^y and OPT_i^z analogously.

Slightly abusing of notation, we write $N_{\mathrm{opt}} \cap C_i^x$ for the set $\{s \cap C_i^x \mid s \in N_{\mathrm{opt}}\}$ of line segments of N_{opt} intersected with 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

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

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 4.

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

Proof. First note that it suffices to consider up-patching; the down-patching case can be argued analogously.

Lemma 7 shows the existence of a near-optimal M-network that up-patches all relevant cuboids. Lemma 8 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.

We argued in Section 2.3 that there exists 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 $O(n^\varepsilon)\mathrm{OPT}$. $\qquad\square$

We now turn to the two lemmas that we just used in the proof of Lemma 6. For our analysis, we need the network N' that is the union of $\mathcal{G}$ with N_{opt} and the projections of N_{opt} onto every separating plane of $\mathcal{G}$. Since there are $3(c-1)$ separating planes and, as we have seen above, $\|\mathcal{G}\| \leq 3c^2\mathrm{OPT}$, it holds that $\|N'\| \leq 3(c^2 + c)\mathrm{OPT} = O(\mathrm{OPT})$.

Lemma 7. *There exists an M-network of total cost at most $3(c^2 + c)\mathrm{OPT}$ that up-patches all relevant cuboids.*

Proof. We claim that N' up-patches all relevant cuboids. To this end, let $t \in C_{ijk}$ and let $t' \in C_{i'j'k'}$ with $i < i', j < j', k < k'$. Follow the M-path connecting t and t', starting from t. This path must leave C_{ijk} at a certain point $\bar{p}$, which lies on some face F of C_{ijk}. Face F, in turn, lies on some separating plane S of the grid $\mathcal{G}$. From now on follow the projection of the M-path from $\bar{p}$ to t' on

13

plane S. This projected path must leave the face F, since t' lies in $C_{i'j'k'}$ with $i < i', j < j', k < k'$, and the projection of t' onto S must therefore lie outside of F. Moreover, the point $\bar{p}'$ where this path leaves F must lie on an edge of C_{ijk} incident to p_{ijk}. Hence, we obtain a t–p_{ijk} M-path by going from t to $\bar{p}$, from $\bar{p}$ to $\bar{p}'$ and then from $\bar{p}'$ to p_{ijk}. □

Lemma 8. *Given a number $\rho \geq 1$ and an efficient ρ-approximation of 3D-RSA, we can efficiently up-patch all relevant cuboids at cost no more than $12(c^2 + c)\rho\mathrm{OPT}$.*

Proof. In Lemma 7, we showed the existence of a network N' that up-patches all relevant cuboids at low cost. Now consider an arbitrary relevant cuboid C_{ijk}. Clearly $N' \cap C_{ijk}$ up-patches C_{ijk}. Hence $\mathrm{OPT}_{ijk}^{\mathrm{up}} \leq \|N' \cap C_{ijk}\|$, where $\mathrm{OPT}_{ijk}^{\mathrm{up}}$ denotes the cost of a minimum up-patching of C_{ijk}. The problem of optimally up-patching C_{ijk} is just an instance I_{ijk} of 3D-RSA in which all terminals in C_{ijk} have to be connected by an M-path to $\hat{p}_{ijk}$. Applying the factor-ρ approximation algorithm for 3D-RSA to each instance I_{ijk} with C_{ijk} relevant, we patch at total cost at most

$$\rho \sum_{C_{ijk} \text{ relevant}} \mathrm{OPT}_{ijk}^{\mathrm{up}} \quad \leq \quad \rho \sum_{C_{ijk} \text{ relevant}} \|N' \cap C_{ijk}\| \quad \leq \quad 4\rho\|N'\|\,.$$

The last inequality follows from the fact that each edge of N' occurs in at most *four* cuboids. The lemma follows since $\|N'\| \leq 3(c^2 + c)\mathrm{OPT}$. □

Running time. 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.

This completes the proof of Theorem 4.

5 Open Problems

We have presented, for any $\varepsilon > 0$, a grid-based $O(n^\varepsilon)$-approximation algorithm 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. For 3D, we have described a $4(k-1)$-approximation algorithm for the case when the terminals lie on $k \geq 2$ horizontal planes. This outperforms our grid-based algorithm when $k \in o(n^\varepsilon)$. Whereas 2D-MMN admits a 2-approximation [5,10,15], it remains open whether $O(1)$- or $O(\log n)$-approximation algorithms exist for 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 better approximation ratios. 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 3) is still unknown. Currently, the best approximation has a ratio of 4, which is (trivially) implied by the result of Soto and Telha [17]. Any progress would immediately improve the approximation ratio of our algorithm for the k-plane case of 3D-MMN (for any $k > 2$).

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 Sympos. Theory Comput. (STOC'95), pp. 489–498 (1995). URL http://dx.doi.org/10.1145/225058.225191
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). URL http://dx.doi.org/10.1016/j.comgeo.2005.09.004
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'98), 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). URL http://dx.doi.org/10.1016/j.tcs.2007.10.013
6. Chin, F., Guo, Z., Sun, H.: Minimum Manhattan network is NP-complete. Discrete Comput. Geom. **45**, 701–722 (2011). URL http://dx.doi.org/10.1007/s00454-011-9342-z
7. Feldman, M., Kortsarz, G., Nutov, Z.: Improved approximating algorithms for directed Steiner forest. In: Proc. 20th ACM-SIAM Sympos. Discrete Algorithms (SODA'09), pp. 922–931 (2009)
8. Fuchs, B., Schulze, A.: A simple 3-approximation of minimum Manhattan networks. Tech. Rep. 570, Zentrum für Angewandte Informatik Köln (2008). URL http://e-archive.informatik.uni-koeln.de/570
9. Gudmundsson, J., Levcopoulos, C., Narasimhan, G.: Approximating a minimum Manhattan network. Nordic J. Comput. **8**, 219–232 (2001)
10. Guo, Z., Sun, H., Zhu, H.: Greedy construction of 2-approximation minimum Manhattan network. In: S. Hong, H. Nagamochi, T. Fukunaga (eds.) Proc. 19th Annu. Internat. Sympos. Algorithms Comput. (ISAAC'08), *Lecture Notes Comput. Sci.*, vol. 5369, pp. 4–15. Springer, Berlin (2008). URL http://dx.doi.org/10.1007/978-3-540-92182-0_4
11. Kato, R., Imai, K., Asano, T.: An improved algorithm for the minimum Manhattan network problem. In: P. Bose, P. Morin (eds.) Proc. 13th Annu. Internat. Sympos. Algorithms Comput. (ISAAC'02), *Lecture Notes Comput. Sci.*, vol. 2518, pp. 344–356. Springer, Berlin (2002). URL http://link.springer.de/link/service/series/0558/bibs/2518/25180344.htm
12. Lam, F., Alexandersson, M., Pachter, L.: Picking alignments from (Steiner) trees. J. Comput. Biol. **10**, 509–520 (2003)
13. Lu, B., Ruan, L.: Polynomial time approximation scheme for the rectilinear Steiner arborescence problem. J. Comb. Optim. **4**(3), 357–363 (2000)
14. Muñoz, X., Seibert, S., Unger, W.: The minimal Manhattan network problem in three dimensions. In: S. Das, R. Uehara (eds.) Proc. 3rd Internat. Workshop Algorithms Comput. (WALCOM'09), *Lecture Notes Comput. Sci.*, vol. 5431, pp. 369–380. Springer, Berlin (2009). URL http://dx.doi.org/10.1007/978-3-642-00202-1_32
15. Nouioua, K.: Enveloppes de Pareto et réseaux de Manhattan: Caractérisations et algorithmes. Ph.D. thesis, Université de la Méditerranée (2005). URL http://www.lif-sud.univ-mrs.fr/~karim/download/THESE_NOUIOUA.pdf
16. Seibert, S., Unger, W.: A 1.5-approximation of the minimal Manhattan network problem. In: X. Deng, D. Du (eds.) Proc. 16th Intern. Symp. Algorithms Comput. (ISAAC'05), *Lecture Notes Comput. Sci.*, vol. 3827, pp. 246–255. Springer, Berlin (2005). URL http://dx.doi.org/10.1007/11602613_26
17. Soto, J.A., Telha, C.: Jump number of two-directional orthogonal ray graphs. In: O. Günlük, G. Woeginger (eds.) Proc. 16th Internat. Conf. Integer Prog. Comb. Optimization (IPCO'11), *Lecture Notes Comput. Sci.*, vol. 6655, pp. 389–403. Springer, Berlin (2011). URL http://dx.doi.org/10.1007/978-3-642-20807-2_31
18. Zelikovsky, A.: A series of approximation algorithms for the acyclic directed Steiner tree problem. Algorithmica **18**(1), 99–110 (1997)

Appendix: Extension to Higher Dimensions

We now describe the approximation algorithm for dD-MMN, for $d > 3$, as a generalization of the 3D-MMN idea from Section 4.

Theorem 5. *For any fixed dimension d and for any $\varepsilon > 0$, there exists an $O(n^\varepsilon)$-approximation algorithm for dD-MMN.*

Large parts of the algorithm and the analysis are straightforward generalizations of the algorithm for 3D-MMN. The presentation of both follows closely the 3D case. However, Lemma 11, where the cost of the patching procedure is bounded, requires non-trivial additional insights.

As in the 3D case we decompose the overall problem into a constant number of instances of the directional subproblem. The directional subproblem consists in M-connecting any terminal pair (t, t') such that $x^i(t) \leq x^i(t')$ for any $i \in \{1, \ldots, d\}$. Here, we use $x^i(p)$ to denote the i-th coordinate of point $p \in \mathbb{R}^d$. We can decompose the general problem into 2^{d-1} directional subproblems. Once again, we assume that the terminals are in general position.

The dD Grid Algorithm

We begin the description with a high-level summary. To solve the directional dD-MMN problem we place a dD-grid which partitions the instance into cuboids and slabs. Terminal pairs lying in different slabs are handled by M-connecting each terminal to the corner of its cuboid and then using the edges of grid. Terminal pairs from the same slab are M-connected by applying the algorithm recursively to all slabs. Each slab contains only a constant fraction of the terminals.

Step 1: Partitioning into cuboids and slabs. Consider the bounding cuboid C for the set T of terminals and choose a large constant $c = d^{1/\varepsilon}$. For each dimension $i \in \{1, \ldots, d\}$ we choose $c + 1$ separating planes determined by values $x_1^i < \cdots < x_{c+1}^i$. Planes x_1^i and x_{c+1}^i coincide with the boundary of C in dimension i. The separating planes for dimension i partition C into c slabs C_j^i, where $j \in \{1, \ldots, c\}$. Slab C_j^i is the set of points $p \in C$ such that $x_i^j \leq x_i(p) \leq x_i^{j+1}$. We place the separating planes so that each slab contains at most n/c terminals. Altogether we have $d(c + 1)$ separating planes.

Let $j_1, \ldots, j_d \in \{1, \ldots, c\}$. The *subcuboids* $C_{j_1, \ldots, j_d}$ is the set of points $p \in C$ such that $x_{j_i}^i \leq x^i(p) \leq x_{j_i+1}^i$ for each $i \in \{1, \ldots, d\}$.

Consider the ith dimension, $i \in \{1, \ldots, d\}$ and integers $j_k \in \{2, \ldots, c\}$ for each $k \in \{1, \ldots, d\} - \{i\}$. Let s be the axis-parallel line segment that contains all points $p \in C$ such that $x^k(p) = x_{j_k}$ for each $k \neq i$. We call s a *grid segment for dimension i*. The *grid* $\mathcal{G}$ is the set of all grid segments and there are $d(c-1)^{d-1}$ grid segments in $\mathcal{G}$.

Given $j_k \in \{2, \ldots, c\}$ for each $k \in \{1, \ldots, d\}$ we call $(x_{j_1}^1, \ldots, x_{j_d}^d)$ a *grid point* of $\mathcal{G}$ and there are $(c-1)^d$ grid points in total.

Step 2: Add M-paths between different slabs. Consider two cuboids $C_{j_1, \ldots, j_d}$ and $C_{j_1', \ldots, j_d'}$ with $j_i < j_i'$ for each $i \in \{1, \ldots, d\}$. Any pair of terminals $t \in C_{j_1', \ldots, j_d'}$ and $t' \in C_{j_1', \ldots, j_d'}$ can be M-connected using the segments of $\mathcal{G}$ as long as t and t' are suitably connected to the corners (grid points) of their cuboids. We use *patching* (described below) to connect all terminal to the corners of their cuboid.

Patching: Call a cuboid $C_{j_1, \ldots, j_d}$ *relevant* if there is a cuboid $C_{j_1', \ldots, j_d'}$ that contains at least one terminal and satisfies $j_i < j_i'$ for each $i \in \{1, \ldots, d\}$. For each relevant cuboid $C_{j_1, \ldots, j_d}$, let $p_{j_1, \ldots, j_d}$

16

denote the grid point $(x^i_{j_1+1}, \dots, x^d_{j_d+1})$. *Up-patching* $C_{j_1,\dots,j_d}$ means to M-connect every terminal in $t \in C_{j_1,\dots,j_d}$ to $p_{j_1,\dots,j_d}$. Up-patch $C_{j_1,\dots,j_d}$ by solving the dD-RSA problem with the terminals inside $C_{j_1,\dots,j_d}$ as the terminals and $p_{j_1,\dots,j_d}$ as the origin.

Down-patching is defined analogously; cuboid $C_{j_1,\dots,j_d}$ is relevant if there is a non-empty cuboid $C_{j'_1,\dots,j'_d}$ with $j'_i < j_i$, $i = 1,\dots,d$ and using grid point $p'_{j_1,\dots,j_d} := (x^i_{j_1+1}, \dots, x^d_{j_d+1})$ as origin instead of $p_{j_1,\dots,j_d}$.

The output of this step is the union of grid $\mathcal{G}$ with a network that up-patches and down-patches all relevant cuboids. This produces M-paths between all terminal pairs in different slabs.

Step 3: Add M-paths within slabs. To also connect terminal pairs that lie in a common slab we apply the algorithm (Steps 1–3) recursively to each slab C^i_j with $i \in \{1,\dots,d\}$ and any $j \in \{1,\dots,c\}$.

Analysis

We now show that the algorithm presented above yields a feasible solution to directional dD-MMN, with cost at most $O(n^\varepsilon)$OPT, for any $\varepsilon > 0$. Here, OPT denotes the cost of an optimum solution to the *general* dD-MMN instance rather than the minimum cost OPT$'$ achievable for the directional subproblem. The reason is that the cost of the grid $\mathcal{G}$ is generally not related to OPT$'$ but to OPT. We finish the section by arguing that the running time of the algorithm is $n^{O(1/\varepsilon)}$.

Lemma 9 (Feasibility). *The dD grid algorithm M-connects all relevant terminal pairs.*

Proof. Let (t, t') be a relevant terminal pair. First suppose that $t \in C_{j_1,\dots,j_d}$ and $t' \in C_{j'_1,\dots,j'_d}$ where $j_i < j'_i$ for all $i \in \{1,\dots,d\}$. Hence, $C_{j_1,\dots,j_d}$ and $C_{j'_1,\dots,j'_d}$ are relevant for up-patching and down-patching, respectively. Consider the corners $p_{j_1,\dots,j_d}$ of $C_{j_1,\dots,j_d}$ and the corner $p'_{j'_1,\dots,j'_d}$ of $C_{j'_1,\dots,j'_d}$. In the up-patching step of our algorithm we solve an RSA problem with terminals of $C_{j_1,\dots,j_d}$ as the input points, and corner $p_{j_1,\dots,j_d}$ as the origin. By definition, an RSA solution M-connects t to $p_{j_1,\dots,j_d}$. Similarly, down-patching M-connects t' to $p'_{j'_1,\dots,j'_d}$. It follows that t and t' are connected, since $p_{j_1,\dots,j_d}$ and $p'_{j'_1,\dots,j'_d}$ are M-connected via grid $\mathcal{G}$. Both terminals are even M-connected since additionally $t \leq p_{j_1,\dots,j_d} \leq p'_{j'_1,\dots,j'_d} \leq t'$, where $\leq$ denotes the domination relation between points.

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

Approximation ratio. Let $r(n)$ denote the approximation ratio of our algorithm where n is the number of terminals in T. The total cost of our solution consists of the cost for the grid $\mathcal{G}$, the cost of up-patching and down-patching all relevant cuboids, and the cost for the recursive treatment of the slabs in all d dimensions. We analyze each of these costs separately.

The grid $\mathcal{G}$ consists of the $d(c-1)^{d-1}$ grid segments. The length of any grid segment s is a lower bound on OPT. This holds because there are two terminals on the boundary of C whose L_1-distance is at least the length of s. It follows that the cost of the grid is bounded by $d(c-1)^{d-1}$OPT.

Let $r_{\text{patch}}(n)$ denote the cost of patching all relevant cuboids as is done in Step 2. Lemma 10 (given below) proves that $r_{\text{patch}}(n) = O(n^\varepsilon)$OPT.

Now consider the recursive application of the algorithm to all slabs C^i_j, where $i \in \{1,\dots,d\}$ and $j \in \{1,\dots,c\}$. First recall that we placed the separating planes so that $|C^i_j| \leq n/c$ for any $i \in \{1,\dots,d\}$ and any $j \in \{1,\dots,c\}$.

Consider dimension $i \in \{1, \ldots, d\}$. Let OPT_j^i be the optimum cost for M-connecting *all* (not only relevant) terminal pairs in slab C_j^i, where $j \in \{1, \ldots, c\}$. Slightly abusing notation, we write $N_{\mathrm{opt}} \cap C_j^i$ for the set of line segments of N_{opt} that are completely contained in the slab C_j^i. Observe that $N_{\mathrm{opt}} \cap C_j^i$, forms a feasible solution for C_j^i. Thus $\mathrm{OPT}_j^i \leq \|N_{\mathrm{opt}} \cap C_j^i\|$. Each such C_j^i contains at most n/c terminals, and therefore the total cost of the solutions for the all slabs C_j^i of dimension i is at most

$$\sum_{j=1}^{c} r\left(\frac{n}{c}\right) \mathrm{OPT}_j^i \leq r\left(\frac{n}{c}\right) \sum_{j=1}^{c} \|N_{\mathrm{opt}} \cap C_j^i\| \leq r\left(\frac{n}{c}\right) \mathrm{OPT}.$$

Summing all costs, we obtain the following recursive equation for $r(n)$

$$r(n)\mathrm{OPT} \leq dc^{d-1} \cdot \mathrm{OPT} + d \cdot r\left(\frac{n}{c}\right) \mathrm{OPT} + r_{\mathrm{patch}}(n)\mathrm{OPT}.$$

Hence $r(n) = O(n^{\max\{\varepsilon, \log_c d\}})$. Choosing $c \geq d^{1/\varepsilon}$, as in Step 1, yields $O(n^\varepsilon)$ proving the approximation ratio claimed in Theorem 4.

Lemma 10. *The cost of patching all relevant cuboids, $r_{\mathrm{patch}}(n)$, is $O(n^\varepsilon)\mathrm{OPT}$.*

Proof. First consider up-patching. Lemma 11 (below) shows the existence of a near optimal network that up-patches all relevant cuboids. Lemma 12 shows that by reducing the patching problem to dD-RSA, we can find such a network of cost $O(\rho)\mathrm{OPT}$, where ρ is the approximation factor of dD-RSA.

Analogously to the 3D-case there is a approximation-preserving reduction from dD-RSA to DST (see Section 2.3), which implies that dD-RSA is approximable within a factor $O(n^\varepsilon)$ for any $\varepsilon > 0$. Hence the same approximation factor can be achieved for dD-RSA by choosing ϵ sufficiently small.

The lemma follows as the analysis holds analogously for down-patching. □

Lemma 11. *There exists an M-network of total cost at most $(c+1)^d\mathrm{OPT}$ that up-patches all relevant cuboids.*

Proof. Let $I \subseteq \{1, \ldots, d\}$ be a set of dimensions. For every $i \in I$ we choose a separating plane $x_{j_i}^i$ where $j_i \in \{1, \ldots, c\}$. Let J be the set of these separating planes and let $C(J)$ be the intersection of C with all separating planes in J. We call $C(J)$ a *C-face*. There are most $(c+1)^d$ such C-faces. Project N_{opt} onto each C-face. Let N' be the union of all these projections. Clearly, the cost of N' is at most $(c+1)^d\mathrm{OPT}$.

We claim that N' up-patches all relevant cuboids. To this end, let (t, t') be a relevant terminal pair such that $t \in C_{j_1, \ldots, j_d}$, $t' \in C_{j_1', \ldots, j_d'}$ and $j_i < j_i'$ for all $i \in \{1, \ldots, d\}$.

We claim that there is an M-path π_t from t to $p_{j_1, \ldots, j_d}$ in N'. To see this, traverse the M-path $\pi_{tt'}$ in N_{opt} connecting t and t', starting from t. We assume w.l.o.g. that the separating planes that bound cuboid $C_{j_1, \ldots, j_d}$ are entered by $\pi_{tt'}$ in the order $(x_{j_1+1}^1, \ldots, x_{j_d+1}^d)$. The desired path π_t starts at t and follows π until the separating plane $x_{j_1+1}^1$ is entered. From this point on we follow the projection $\pi_{tt'}(x_{j_1+1}^1)$ of $\pi_{tt'}$ onto C-face $C(x_{j_1+1}^1)$. If $\pi_{tt'}(x_{j_1+1}^1)$ enters $x_{j_2+1}^2$ we follow the projection $\pi_{tt'}(x_{j_1+1}^1, x_{j_2+1}^2)$ of $\pi_{tt'}$ onto $C(x_{j_1+1}^1, x_{j_2+1}^2))$. We proceed in this fashion until we reach the C-face $C(x_{j_1+1}^1, \ldots, x_{j_d+1}^d)$, which is just the corner $p_{j_1, \ldots, j_d}$. Since N' contains the projection of $\pi_{tt'}$ onto each C-face, the path π_t described above is contained in N'. This reasoning remains valid if the separating planes that bound $C_{j_1, \ldots, j_d}$ are entered in an arbitrary order, as we projected N_{opt} onto each C-face. □

Lemma 12. *Given an efficient algorithm that approximates dD-RSA within a factor of ρ, we can efficiently up-patch all relevant cuboids at cost at most $(2(c+1))^d \rho \mathrm{OPT}$.*

Proof. In Lemma 11, we showed the existence of a network N' that up-patches all relevant cuboids at low cost. Now consider an arbitrary relevant cuboid $C_{j_1,\ldots,j_d}$. Clearly $N' \cap C_{j_1,\ldots,j_d}$ up-patches $C_{j_1,\ldots,j_d}$. Hence $\mathrm{OPT}^{\mathrm{up}}_{j_1,\ldots,j_d} \leq \|N' \cap C_{j_1,\ldots,j_d}\|$, where $\mathrm{OPT}^{\mathrm{up}}_{j_1,\ldots,j_d}$ denotes the cost of a minimum up-patching of $C_{j_1,\ldots,j_d}$. The problem of optimally up-patching $C_{j_1,\ldots,j_d}$ is just an instance $I_{j_1,\ldots,j_d}$ of dD-RSA, in which all terminals in $C_{j_1,\ldots,j_d}$ have to be connected by an M-path to $p_{j_1,\ldots,j_d}$. Applying the factor-ρ approximation algorithm for dD-RSA to each instance $I_{j_1,\ldots,j_d}$ with $C_{j_1,\ldots,j_d}$ relevant, we patch at total cost at most

$$
\rho \sum_{C_{j_1,\ldots,j_d} \text{ relevant}} \mathrm{OPT}^{\mathrm{up}}_{j_1,\ldots,j_d} \quad \leq \quad \rho \sum_{C_{j_1,\ldots,j_d} \text{ relevant}} \|N' \cap C_{j_1,\ldots,j_d}\| \quad \leq \quad 2^d \rho \|N'\| .
$$

The last inequality follows from the fact that each segment of N' occurs in at most 2^d cuboids. The lemma follows since $\|N'\| \leq (c+1)^d \mathrm{OPT}$. □

Running time. Let $T(n)$ denote the running time of the algorithm for n terminals. The running time is dominated by patching and the recursive treatment of slabs. Using the DST algorithm of Charikar et al. [4], patching cuboid C^i_j requires time $(n^i_j)^{O(1/\varepsilon)}$, where n^i_j is the number of terminals in C^i_j. As each cuboid is patched at most twice and there are c^d cuboids, patching requires total time $O(c^d)n^{O(1/\varepsilon)} = n^{O(1/\varepsilon)}$. The algorithm is applied recursively to dc slabs. This yields the recurrence $T(n) = dcT(n/c) + n^{O(1/\varepsilon)}$, which leads to the claimed running time.