Facility Location and the Geometric Minimum-Diameter Spanning Tree

Joachim Gudmundsson [a,1] Herman Haverkort [b,2]
Sang-Min Park [c] Chan-Su Shin [d,3] Alexander Wolff [e,4]

[a] *Dept. of Comp. Science, Eindhoven University, The Netherlands. Email: h.j.gudmundsson@tue.nl*

[b] *Dept. of Comp. Science, Utrecht University, The Netherlands. Email: herman@cs.uu.nl*

[c] *Dept. of Comp. Science, KAIST, Korea. Email: smpark@jupiter.kaist.ac.kr*

[d] *School of Electr. and Inform. Engineering, Hankuk University of Foreign Studies, Korea. Email: cssin@hufs.ac.kr*

[e] *Institute for Logic, Complexity and Deduction Systems, Karlsruhe University, Germany. Internet: http://i11www.ilkd.uni-karlsruhe.de/~awolff*

Abstract

Let P be a set of n points in the plane. The geometric minimum-diameter spanning tree (MDST) of P is a tree that spans P and minimizes the Euclidian length of the longest path. It is known that there is always a mono- or a dipolar MDST, i.e. a MDST with one or two nodes of degree greater 1, respectively. The more difficult dipolar case can so far only be solved in slightly subcubic time.

This paper has two aims. First, we present a solution to a new data structure for facility location, the minimum-sum dipolar spanning tree (MSST), that mediates between the minimum-diameter dipolar spanning tree and the discrete two-center problem (2CP) in the following sense: find two centers p and q in P that minimize the sum of their distance plus the distance of any other point (client) to the closer center. This is of interest if the two centers do not only serve their customers (as in the case of the 2CP), but frequently have to exchange goods or personnel between themselves. We give an $O(n^2 \log n)$-time algorithm for this problem. A slight modification of our algorithm yields a factor-4/3 approximation of the MDST.

Second, we give two fast approximation schemes for the MDST, i.e. factor-$(1+\varepsilon)$ approximation algorithms. One uses a grid and takes $O^*(E^{6-1/3}+n)$ time, where $E = 1/\varepsilon$ and the O^*-notation hides terms of type $O(\log^{O(1)} E)$. The other uses the well-separated pair decomposition and takes $O(nE^3 + En \log n)$ time. A combination of the two approaches runs in $O^*(E^5 + n)$ time. Both schemes can also be applied to MSST and 2CP.

Preprint submitted to Elsevier Science *19 February 2004*

1 Introduction

The MDST can be seen as a network without cycles that minimizes the maximum travel time between any two sites connected by the network. This is of importance, e.g. in communication systems where the maximum delay in delivering a message is to be minimized. Ho et al. showed there is always a mono- or a dipolar MDST [HLCW91]. For a different proof, see [HT95]. Ho et al. also gave an $O(n \log n)$-time algorithm for the monopolar and an $O(n^3)$-time algorithm for the dipolar case [HLCW91]. In addition, they showed that the problem becomes considerably easier when allowing Steiner points, i.e. to find a spanning tree with minimum diameter over all point sets P' that contain the input point set P. The reason is that there always is a minimum-diameter Steiner tree that is monopolar and whose pole is the center of the smallest enclosing circle of P. Thus the minimum-diameter Steiner tree can be determined in linear time [HLCW91].

The cubic time bound for the dipolar case was recently improved by Chan [Cha02] to $\tilde{O}(n^{3-c_d})$, where $c_d = 1/((d+1)(\lceil d/2 \rceil + 1))$ is a constant that depends on the dimension d of the point set and the $\tilde{O}$-notation hides factors that are $o(n^{\varepsilon})$ for any fixed $\varepsilon > 0$. In the planar case $c_d = 1/6$. Chan speeds up the exhaustive-search algorithm of Ho et al. by using new semi-dynamic data structures. Note however that c_d tends to 0 with increasing d, while the asymptotic running time of the algorithm of Ho et al. does not depend on the dimension.

Note that in the dipolar case the objective is to find the two poles $x, y \in P$ of the tree such that the function $r_x + |xy| + r_y$ is minimized, where $|xy|$ is the Euclidean distance of x and y, and r_x and r_y are the radii of two disks centered at x and y whose union covers P. On the other hand the *discrete k-center problem* is to determine k points in P such that the union of k congruent disks centered at the k points covers P and the radius of the disks is minimized. This is a typical facility location problem: there are n supermarkets and in k of them a regional director must be placed such that the maximum director-supermarket distance is minimized. This problem is NP-hard provided that k is part of the input [GJ79]. Thus, the main research on this problem has focused on small k, especially on $k = 1, 2$. For $k = 1$, the problem can be solved in $O(n \log n)$ time using the farthest-point Voronoi diagram of P. For $k = 2$, the problem becomes considerably harder. Using the notation from above, the

[1] Supported by the Swedish Foundation for International Cooperation in Research and Higher Education.
[2] Supported by the Netherlands' Organization for Scientific Research.
[3] Supported by grant No. R05-2002-000-00780-0 from the Basic Research Program of the Korea Science and Engineering Foundation (KOSEF).
[4] Supported by the KOSEF program Brain Korea 21.

discrete two-center problem consists of finding two centers $x, y \in P$ such that the function $\max\{r_x, r_y\}$ is minimized. Agarwal et al. [ASW98] gave the first subquadratic-time algorithm for this problem. It runs in $O(n^{4/3} \log^5 n)$ time.

In this paper we are interested in (a) a new facility location problem that mediates between the minimum-diameter dipolar spanning tree (MDdST) and the two-center problem and (b) fast approximations of the computationally expensive MDdST. As for our first aim we observe the following. Whereas the MDdST minimizes $|xy| + (r_x + r_y)$, the discrete two-center problem is to minimize $\max\{r_x, r_y\}$, which means that the distance between the two centers is not considered at all. If, however, the two centers need to communicate with each other for cooperation, then their distance should be considered as well—not only the radius of the two disks. Therefore our aim is to find two centers x and y that minimize $|xy| + \max\{r_x, r_y\}$, which is a compromise between the two previous objective functions. We will refer to this problem as the *discrete minimum-sum two-center problem* and call the resulting graph the *minimum-sum dipolar spanning tree* (MSST). As it turns out, our algorithm for the MSST also constitutes a compromise, namely in terms of runtime between the subcubic-time MDdST-algorithm and the superlinear-time 2CP-algorithm. More specifically, in Section 2 we will describe an algorithm that solves the discrete minimum-sum two-center problem in the plane in $O(n^2 \log n)$ time using $O(n^2)$ space. For dimension $d < 5$ a variant of our algorithm is faster than the more general $\tilde{O}(n^{3-c_d})$-time MDST-algorithm of Chan [Cha02] that can easily be modified to compute the MSST instead.

In Section 3 we turn to our second aim, approximations for the MDST. We combine a slight modification of the MSST with the minimum-diameter monopolar spanning tree (MDmST). We identify two parameters that depend on the MDdST and help to express a very tight estimation of how well the two trees approximate it. It turns out that at least one of them is a factor-4/3 approximation of the MDST.

Finally, in Section 4 we show that there are even strong linear-time approximation schemes (LTAS) for the MDST, i.e. algorithms that given a set P of n points and some $\varepsilon > 0$ compute in $O^*(E^c + n)$ time a spanning tree whose diameter is at most $(1 + \varepsilon)$ times as long as the diameter of a MDST. In the runtime expression $E = 1/\varepsilon$, c is a constant and the O^*-notation hides terms of type $O(\log^{O(1)} E)$. The existence of a strong LTAS for the MDST has independently been proven by Spriggs at al. [SKB+03]. Their LTAS is of order $c = 3$, i.e. it takes $O^*(E^3 + n)$ time.

Our results are as follows. Our first LTAS uses a grid of $O(E) \times O(E)$ square cells and runs Chan's exact algorithm [Cha02] on one representative point per cell. The same idea has been used before [BHP99,Cha00] to approximate the diameter of a point set, i.e. the longest distance between any pair of the given

points. Our first LTAS is of order $5\frac{2}{3}$.

Our second approximation scheme is based on the well-separated pair decomposition [CK95] of P and takes $O(E^3n + En\log n)$ time. The well-separated pair decomposition will help us to limit our search for the two poles of an approximate MDdSTto a linear number of point pairs. If we run our second scheme on the $O(E^2)$ representative points in the grid mentioned above, we get a LTAS of order 5. Both schemes can be adjusted to approximate the MSST and the 2CP within the same time bounds.

We will refer to the diameter d_P of the MDST of P as the *tree diameter* of P. We assume that P contains at least four points.

2 The Minimum-Sum Dipolar Spanning Tree

It is simple to give an $O(n^3)$-time algorithm for computing the MSST. Just go through all $O(n^2)$ pairs $\{p, q\}$ of input points and compute in linear time the point m_{pq} whose distance to the current pair is maximum. In order to give a faster algorithm for computing the MSST, we need a few definitions. Let h_{pq} be the closed halfplane that contains p and is delimited by the perpendicular bisector b_{pq} of p and q. Note that $h_{pq} \cap h_{qp} = b_{pq} = b_{qp}$. Let $\mathcal{T}_{pq}$ be the tree with dipole $\{p, q\}$ where all other points are connected to the closer pole. (Points on b_{pq} can be connected to either p or q.) Clearly the tree $\mathcal{T}_{pq}$ that minimizes $|pq| + \min\{|pm_{pq}|, |qm_{pq}|\}$ is an MSST. The following two observations will speed up the MSST computation.

We first observe that we can split the problem of computing all points of type m_{pq} into two halves. Instead of computing the point m_{pq} farthest from the pair $\{p, q\}$, we compute for each ordered pair (p, q) a point $f_{pq} \in P \cap h_{pq}$ that is farthest from p. See Figure 1 for an example. Now we want to find the tree $\mathcal{T}_{pq}$ that minimizes $|pq| + \max\{|pf_{pq}|, |qf_{qp}|\}$. We will see that other than the points of type m_{pq} we can compute those of type f_{pq} in batch.

Our algorithm consists of two phases, see Algorithm 1. In phase I we go through all points p in P. The central (and time-critical) part of our algorithm is the procedure ComputeAllFarthest that computes f_{pq} for all $q \in P \setminus \{p\}$. For a trivial $O(n^2)$-time implementation of this precedure, see Algorithm 2. In phase II we then use the above form of our target function to determine the MSST.

The second important observation that helped us to speed up ComputeAllFarthest is the following. Let p be fixed. Instead of computing f_{pq} for each $q \in P \setminus \{p\}$ individually, we characterize in Lemma 1 all q that have the same

Algorithm 1 MSST(P)

Phase I: compute all f_{pq}
  for each $p \in P$ **do**
    ComputeAllFarthest(P, p)
  end for p
Phase II: search for MSST
  for each $\{p, q\} \in \binom{P}{2}$ **do**
    $d_{pq} \leftarrow |pq| + \max\{|pf_{pq}|, |qf_{qp}|\}$
  end for $\{p, q\}$
return $\mathcal{T}_{pq}$ with d_{pq} minimum.

Algorithm 2 ComputeAllFarthest(P, p) {first version}

  for each $q \in P \setminus \{p\}$ **do**
    $f_{pq} \leftarrow p$
    for each $r \in P \setminus \{p, q\}$ **do**
      if $r \in h_{pq}$ **and** $|pr| > |pf_{pq}|$ **then**
        $f_{pq} \leftarrow r$
      end if
    end for q
  end for r
  return f_{pq} for each $q \in P \setminus \{p\}$

f_{pq}. Our characterization uses the following direct consequence of Thales' Theorem. See Figure 2 as illustration.

Fact 1 *Let $D(x, p)$ be the open disk that is centered at x and whose boundary contains p ($D(x, p) = \emptyset$ if $x = p$). Then $x \in h_{pq}$ if and only if $q \notin D(x, p)$.*

Lemma 1 *x is farthest from p in $P \cap h_{pq}$ if and only if $q \notin D(x, p)$ and, for all $x' \in P$ with $|px'| > |px|$, $q \in D(x', p)$.*

PROOF. "If" part: Due to $q \notin D(x, p)$ and Fact 1 we know that x lies in h_{pq}. Fact 1 also yields that all $x' \in P$ with $|px'| > |px|$ do *not* lie in h_{pq} since $q \in D(x', p)$ for all such x'. Thus x is farthest from p among all points in h_{pq}.

"Only if" part: suppose $q \in D(x, p)$ or suppose there is an $x' \in P$ with $|px'| > |px|$ and $q \notin D(x', p)$. In the former case we would have $x \notin h_{pq}$, in the latter $|px'| > |px|$ and $x' \in h_{pq}$. Both would contradict x being farthest from p among the points in h_{pq}. □

Lemma 1 immediately yields a way to set the variables f_{pq} in batch: go through the points $q_i \in P \setminus \{p\}$ in order of non-increasing distance from p, find all points in $P_i = P \setminus D(q_i, p)$, set f_{pq} to q_i for all $q \in P_i$, remove the points in P_i

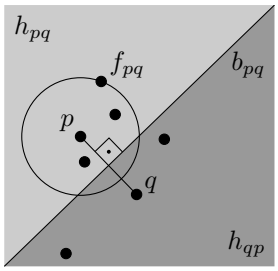

Fig. 1. f_{pq} denotes the point farthest from p in $P \cap h_{pq}$.

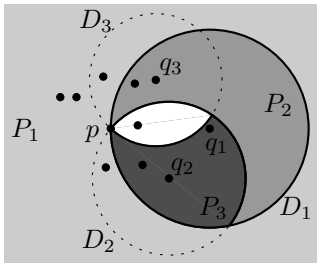

Fig. 2. $x \in h_{pq}$ if and only if $q \notin D(x, p)$.

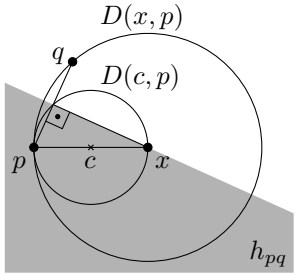

Fig. 3. Computing the points of type f_{pq} in batch.

from P, and continue—see the second version of ComputeAllFarthest in Algorithm 3.

Figure 3 visualizes what happens in the first three runs through the outer for-loop of Algorithm 3: the areas shaded light, medium, and dark contain all points q with $f_{pq} = q_1$, $f_{pq} = q_2$, and $f_{pq} = q_3$, respectively. We used D_i as shorthand for $D(q_i, p)$.

Algorithm 3 ComputeAllFarthest(P, p) {second version}

```
1: sort P = q_1, ..., q_n such that |pq_1| ≥ |pq_2| ≥ ··· ≥ |pq_n|
2: P ← P \ {p}
3: for i ← 1 to n do
4:    P_i ← P \ D(q_i, p)
5:    for each q ∈ P_i do
6:       f_pq ← q_i
7:    end for q
8:    P ← P \ P_i
9: end for i
10: return f_pq for each q ∈ P \ {p}
```

Lemma 2 *For each $q \in P \setminus \{p\}$ Algorithm 3 sets f_{pq} to the point farthest from p in $P \cap h_{pq}$.*

PROOF. Note that $P \setminus \{p\} = \bigcup_{i=1}^{n} P_i$. This is due to the fact that $D(q_n, p) = D(p, p) = \emptyset$. Thus the variables f_{pq} are in fact set for all $q \in P \setminus \{p\}$ in line 6 of Algorithm 3.

The values that are assigned to the f_{pq}'s are correct due to the order, in which the outer for-loop runs through the points in P: f_{pq} is set to q_i if i is the smallest index such that $q \in P_i$. This is the case if i is the smallest index such that $q \notin D(q_i, p)$ and $q \in D(q_j, p)$ for $j < i$. Since the q_j with $j < i$ are exactly

the points in P farther from p than q_i, Lemma 1 yields that q_i is the point farthest from p in $P \cap h_{pq}$. $\square$

The remainder of this section deals with efficiently finding points in P_i. We give two methods. The first, which is slightly slower in the plane, also works for higher dimensions.

Method I. We use dynamic circular range searching, which is a special case of halfspace range searching in $\mathbb{R}^3$ via orthogonal projection to the paraboloid $\{(x, y, z) \mid z = x^2 + y^2\}$ [AM95]. The necessary data structure can be build in $O(n^{1+\varepsilon})$ time and space for an arbitrarily small $\varepsilon > 0$. After each query with the halfspace H_i corresponding to the complement of the disk $D(q_i, p)$ all points in H_i must be deleted (according to step 8 of Algorithm 3). The total time for querying and deleting is $O(n^{1+\varepsilon})$. This yields an $O(n^{2+\varepsilon})$-time algorithm for finding the MSST. We will give a faster algorithm for the planar case. However, it is not clear how that algorithm can be generalized to higher dimensions. For dimensions $d \in \{3, 4\}$ computing the MSST with range searching takes $O(n^{2.5+\varepsilon})$ time [AM95]. This is faster than Chan's MDST-algorithm [Cha02] that can easily be modified to compute the MSST instead. His algorithm runs in $\tilde{O}(n^{3-c_d})$ time, where $c_d = 1/((d+1)(\lceil d/2 \rceil + 1)) \leq 1/12$ for $d \geq 3$.

Method II. We compute a partition of the plane into regions $R_1, \ldots, R_n$ such that R_i contains the set P_i of all points q with $f_{pq} = q_i$. Then we do a plane sweep to determine for each $q \in P \setminus \{p\}$ the region R_i that contains it. Method II takes $O(n \log n)$ time and thus yields an $O(n^2 \log n)$-time algorithm for finding the MSST in the plane. We will use the following simple fact.

Fact 2 *Given a set $\mathcal{D}$ of disks in the plane that all touch a point p, each disk contributes at most one piece to the boundary of $\bigcap \mathcal{D}$.*

This helps us to bound the complexity of our planar partition.

Lemma 3 *Let $\mathcal{D} = \{D_1, \ldots, D_{n-1}\}$ be a set of disks in the plane, let $D_0 = \mathbb{R}^2$, $D_n = \emptyset$, and for $i = 1, \ldots, n$ let $I_i = D_0 \cap \cdots \cap D_{i-1}$ and $R_i = I_i \setminus D_i$. Then $\mathcal{R}(\mathcal{D}) = \{R_1, \ldots, R_n\}$ is a partition of the plane whose complexity—the total number of arcs on the boundaries of $R_1, \ldots, R_n$—is $O(n)$.*

PROOF. The region R_i consists of all points that lie in $D_0, \ldots, D_{i-1}$ but not in D_i. Since $D_0 = \mathbb{R}^2$ and $D_n = \emptyset$ it is clear that $\mathcal{R}(\mathcal{D})$ is in fact a partition of the plane. For an example refer to Figure 4, where the regions R_1, R_2, R_3, and R_4 are shaded from light to dark gray.

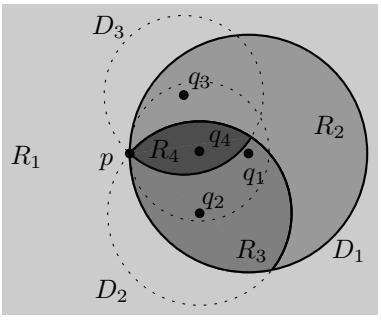

Fig. 4. Regions of the partition $\mathcal{R}(\mathcal{D})$.

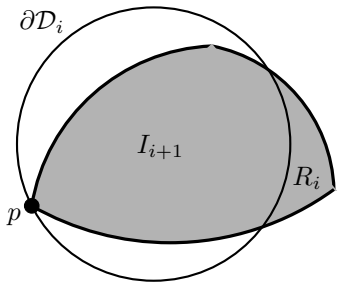

Fig. 5. A step in the incremental construction of $\mathcal{R}(\mathcal{D})$.

If $\mathcal{R}(\mathcal{D})$ is constructed incrementally, each new disk D_i splits I_i into I_{i+1} and R_i. For illustration, see Figure 5, where I_i is the shaded region. Due to Fact 2, D_i contributes at most one circular arc A to the boundary of I_{i+1}. The start- and endpoint of A can split two arcs on the boundary of I_i into two pieces each. Two of these at most four pieces will belong to I_{i+1} and two to R_i. Thus the number of arcs in $\mathcal{R}(\mathcal{D})$ increases at most by three when adding a new disk to the current partition. □

Now we can give a faster implementation of ComputeAllFarthest for the planar case. More specifically we will show the following.

Lemma 4 *Let $D_0 = \mathbb{R}^2$ and $D_n = \emptyset$. Given a set Q of m points and a set $\mathcal{D} = \{D_1, \ldots, D_{n-1}\}$ of disks in the plane that all touch a point p, there is an algorithm that computes in $O((m+n)\log n)$ total time for each point $q \in Q$ the region $R \in \mathcal{R}(\mathcal{D})$ that contains q. The algorithm needs $O(n+m)$ space.*

PROOF. We first construct the partition $\mathcal{R}(\mathcal{D})$ and then do a plane sweep to locate the points in Q in the cells of $\mathcal{R}(\mathcal{D})$.

We use the incremental construction of $\mathcal{R}(\mathcal{D})$ as in Lemma 3. In order to find the two points where the boundary ∂D_i of a new disk D_i intersects the boundary ∂I_i of I_i we need a data structure $\mathcal{T}$ that stores the circular arcs on ∂I_i. The data structure must allow us to do search, remove, insert, and successor operations in logarithmic time. This is standard, e.g. for red-black trees [CLR90]. In $\mathcal{T}$ we store the circular arcs on ∂I_i in clockwise order starting from p. We assume that p is the leftmost point of ∂I_i.

By Fact 2, ∂D_i intersects ∂I_i at zero, one, or two points other than p. Let A and B be the first respectively last arc on ∂I_i (both incident to p), and let A' and B' be infinitesimally small pieces of A respectively B incident to p. There are three cases, which can be distinguished from each other in constant time. For illustration see Figures 6 to 8, where we have sketched ∂I_i as a polygon for simplicity.

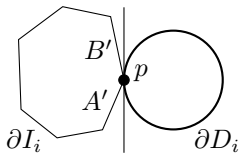

Fig. 6. Case (1).

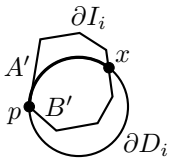

Fig. 7. Case (2).

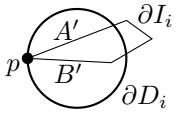

Fig. 8. Case (3).

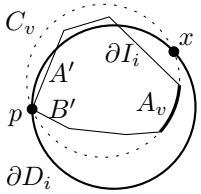

Fig. 9. Querying $\mathcal{T}$ at v.

(1) $D_i \cap I_i = \emptyset$.
This can be verified by checking whether the tangent of D_i in p separates D_i from A and B. If this is the case, we are done since then $R_i = I_i$ and $R_j = I_j = \emptyset$ for all $j > i$.

(2) $D_i \cap I_i \neq \emptyset$, and at least one of A' or B' lies outside I_i.
We follow the part of ∂I_i from p that lies outside D_i until we reach an intersection point x (possibly again p) of ∂D_i and ∂I_i. On our way we replace all nodes in $\mathcal{T}$ that correspond to arcs outside D_i by a new node that corresponds to the arc $\partial D_i \cap I_i$ (bold in Figure 7). The region $R_i = I_i \setminus D_i$ is delimited by the new arc and all arcs on ∂I_i from p up to x.

(3) $D_i \cap I_i \neq \emptyset$, and A' and B' lie inside I_i.
We query $\mathcal{T}$ to find out whether and where ∂D_i and ∂I_i intersect other than in p. We follow a path from the root of $\mathcal{T}$ either to a node whose arc intersects ∂D_i, or to a leaf if ∂I_i and ∂D_i do not intersect. Let v be the current inner node of $\mathcal{T}$, A_v the corresponding arc and C_v the circle that contains A_v (and touches p). We consider the following two subcases:

(a) $\partial D_i \cap C_v = \{p\}$.
This occurs in the degenerate case that p and the centers of D_i and C_v are collinear. If C_v is contained in D_i, then I_i is also contained in D_i. Thus $R_i = \emptyset$, $I_{i+1} = I_i$, and we can continue with D_{i+1}.
Otherwise A_v lies outside D_i—recall that $C_v, D_i \in \mathcal{D}$ and that all disks in $\mathcal{D}$ are pairwise different. Since A' and B' lie inside D_i, ∂D_i intersects two arcs on ∂I_i; one between A' and A_v, and one between A_v and B'. Thus we can continue to search for an intersection in any of the two subtrees of v.

(b) $\partial D_i \cap C_v = \{p, x\}$ and $x \neq p$.
If $x \in A_v$, we stop. Otherwise we continue our search in the left or right subtree of v depending on whether p, x, and A_v lie on C_v in clockwise or counterclockwise order, respectively. Note that in the clockwise case the part of ∂I_i from A_v to B' (in clockwise order) is completely contained in D_i, see Figure 9. Thus no arc in the right subtree of v intersects ∂D_i. The counterclockwise case is symmetric.

If—in case (3b)—we reach a leaf of $\mathcal{T}$ without finding any intersection, this means that $\partial I_i \cap \partial D_i = \{p\}$, since in each step of the query we have only discarded arcs in $\mathcal{T}$ that lie on the portion of ∂I_i that cannot intersect ∂D_i. The fact that A' and B' lie inside D_i now yields $I_i \subseteq D_i$. Thus $R_i = \emptyset$, and we can continue with D_{i+1}.

Otherwise we have found some $x \neq p$ that lies on $\partial D_i \cap \partial I_i$. If it turns out that ∂D_i and ∂I_i just touch in x, then we again have $I_i \subseteq D_i$, which means that $R_i = \emptyset$ and we can proceed with D_{i+1}. If, however, ∂D_i and ∂I_i intersect properly, then we continue as in case (2), following the part of ∂I_i outside D_i from x until we hit a second intersection point x'. Due to Fact 2 there cannot be any further intersection points.

Whenever we modify $\mathcal{T}$ we also do the necessary steps in the incremental construction of $\mathcal{R}(\mathcal{D})$: we create circular pointers around each R_i and pointers from each arc to the two regions it borders.

The plane sweep is practically the same as for locating points in a vertical decomposition of line segments. Our (multi-) set E of event points consists of the set V of vertices of $\mathcal{R}(\mathcal{D})$, the points in Q, and the set X of the left- and rightmost points of arcs that are not arc endpoints. Note that there is a linear order among the arcs in the vertical strip between any two consecutive event points in $V \cup X$. We store the points in E in an array and sort them according to non-decreasing x-coordinate. The sweep-line status consists of the arcs in $\mathcal{R}(\mathcal{D})$ that are currently intersected by the vertical sweep line ℓ in the order in which they are intersected by ℓ. The sweep-line status $\mathcal{S}$ can be implemented by any balanced binary search tree (like a red-black tree) that allows insertion, deletion, and search in logarithmic time.

Each time ℓ hits an event point in X or V, we either add an arc to $\mathcal{S}$ or remove an arc from $\mathcal{S}$. (This assumes non-degeneracy of $\mathcal{D}$, i.e. no three disk boundaries intersect in a point other than p. The assumption can be overcome by using several event points for vertices in $\mathcal{R}(\mathcal{D})$ of degree greater than 3.) Each time ℓ hits an event point $q \in Q$ we determine the first arc in $\mathcal{S}$ above or below q and return the index i of the corresponding region R_i of $\mathcal{R}(\mathcal{D})$.

The data structure for $\mathcal{R}(\mathcal{D})$ can be set up in $O(n \log n)$ time due to Lemma 3: each step of the incremental construction takes $O(\log n)$ time for querying $\mathcal{T}$ and $O(|R_i| \log n)$ time for updating $\mathcal{T}$ and $\mathcal{R}(\mathcal{D})$. Lemma 3 also ensures that E consists of at most $O(n+m)$ points, and that $\mathcal{S}$ contains at most $O(n)$ arcs at any time during the sweep. Since each event point is processed in $O(\log n)$ time, the whole sweep takes $O((m+n)\log n)$ time. □

Now we can conclude:

Theorem 1 *There is an algorithm that computes an MSST in $O(n^2 \log n)$ time using quadratic space.*

PROOF. We can implement ComputeAllFarthest by applying the algorithm of Lemma 4 to $\mathcal{D} = \{D(q_1, p), \ldots, D(q_{n-1}, p)\}$ and $Q = P \setminus \{p\}$. This yields a running time of $O(n \log n)$ for ComputeAllFarthest. With this subroutine, Algorithm 1 computes an MSST in $O(n^2 \log n)$ time. □

3 Approximating the minimum-diameter spanning tree

We first make the trivial observation that the diameter of *any* monopolar tree on P is at most twice as long as the tree diameter d_P of P. We use the following notation. Let $\mathcal{T}_{\text{di}}$ be a fixed MDdST and $\mathcal{T}_{\text{mono}}$ a fixed MDmST of P. The tree $\mathcal{T}_{\text{di}}$ has minimum diameter among those trees with vertex set P in which all but two nodes—the poles—have degree 1. The tree $\mathcal{T}_{\text{mono}}$ is a minimum-diameter star with vertex set P. Let x and y be the poles of $\mathcal{T}_{\text{di}}$, and let $\delta = |xy|$ be their distance. Finally let r_x (r_y) be the length of the longest edge in $\mathcal{T}_{\text{di}}$ incident to x (y, respectively) without taking into account the edge xy. Thus disks of radius r_x and r_y centered at x and y, respectively, cover P. Wlog. we assume $r_x \geq r_y$.

Ho et al. showed that in the dipolar case (i.e. if there is no monopolar MDST), the disk centered at y cannot be contained by the one centered at x. We will need this *stability lemma* below.

Lemma 5 (Stability lemma [HLCW91]) $r_x < \delta + r_y$.

In order to get a good approximation of the MDST, we slightly modify the algorithm for the MSST described in Section 2. After computing the $O(n^2)$ points of type f_{pq}, we go through all pairs $\{p, q\}$ and consider the tree $\mathcal{T}_{pq}$ with dipole $\{p, q\}$ in which each point is connected to its closer dipole. In Section 2 we were searching for a tree of type $\mathcal{T}_{pq}$ that minimizes $|pq|+\max\{|f_{pq}p|, |qf_{qp}|\}$. Now we go through all trees $\mathcal{T}_{pq}$ to find the tree $\mathcal{T}_{\text{bisect}}$ with minimum *diameter*, i.e. the tree that minimizes $|pq| + |f_{pq}p| + |qf_{qp}|$. Note that the only edge in $\mathcal{T}_{pq}$ that crosses the perpendicular bisector of pq is the edge pq itself. This is of course not necessarily true for the MDdST $\mathcal{T}_{\text{di}}$. We will show the following:

Lemma 6 *Given a set P of n points in the plane there is a tree with the following two properties: it can be computed in $O(n^2 \log n)$ time using $O(n^2)$ storage, and its diameter is at most $4/3 \cdot d_P$.*

PROOF. Due to Theorem 1 it suffices to show the approximation factor. We will first compute upper bounds for the approximation factors of $\mathcal{T}_{\text{bisect}}$ and $\mathcal{T}_{\text{mono}}$ and then analyze where the minimum of the two takes its maximum.

For the analysis of $\mathcal{T}_{\text{bisect}}$ consider the tree $\mathcal{T}_{xy}$ whose poles are those of $\mathcal{T}_{\text{di}}$. The diameter of $\mathcal{T}_{xy}$ is an upper bound for that of $\mathcal{T}_{\text{bisect}}$. Let r'_x (r'_y) be the length of the longest edge of $\mathcal{T}_{xy}$ incident to x (y, respectively) without taking into account the edge xy. Note that $r'_x = |xf_{xy}|$ and $r'_y = |yf_{yx}|$.

Now we compare the diameter of $\mathcal{T}_{xy}$ to that of $\mathcal{T}_{\text{di}}$. Observe that $\max\{r'_x, r'_y\} \leq r_x$. This is due to our assumption $r_x \geq r_y$ and to the fact that f_{xy} and f_{yx} have at most distance r_x from both x and y. This observation yields diam $\mathcal{T}_{xy}$ =

$r'_x + \delta + r'_y \;\leq\; 2\max\{r'_x, r'_y\} + \delta \;\leq\; 2r_x + \delta$. Now we define two constants α and β that only depend on $\mathcal{T}_{\text{di}}$. Let

$$\alpha \;=\; \frac{\delta}{r_x + r_y} \quad \text{and} \quad \beta \;=\; \frac{r_x}{r_y}.$$

Note that $\alpha > 0$ and $\beta \geq 1$. Introducing α and β yields

$$\frac{\text{diam } \mathcal{T}_{\text{bisect}}}{\text{diam } \mathcal{T}_{\text{di}}} \;\leq\; \frac{\text{diam } \mathcal{T}_{xy}}{\text{diam } \mathcal{T}_{\text{di}}} \;\leq\; \frac{2r_x + \delta}{r_x + \delta + r_y} \;=\; \frac{\alpha(1+\beta) + 2\beta}{(1+\alpha)(1+\beta)} \;=:\; f_{\text{bisect}}(\alpha, \beta),$$

since $2r_x = 2\beta(r_x + r_y)/(1+\beta)$ and $\delta = \alpha(r_x + r_y)$. The function $f_{\text{bisect}}(\alpha, \beta)$ is an upper bound for the approximation factor that $\mathcal{T}_{\text{bisect}}$ achieves.

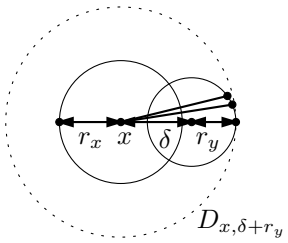

Fig. 10. Approximating $\mathcal{T}_{\text{di}}$ with $\mathcal{T}_{\text{mono}}$.

Now we apply our α-β-analysis to $\mathcal{T}_{\text{mono}}$. The stability lemma $r_x < \delta + r_y$ [HLCW91] implies that all points in P are contained in the disk $D_{x,\delta+r_y}$ of radius $\delta + r_y$ centered at x, see Figure 10. Due to that, the diameter of a monopolar tree $\mathcal{T}$ that spans P and is rooted at x is at most twice the radius of the disk. We know that diam $\mathcal{T}_{\text{mono}} \leq$ diam $\mathcal{T}$ since $\mathcal{T}_{\text{mono}}$ is the MDmST of P. Thus

$$\text{diam } \mathcal{T}_{\text{mono}} \;\leq\; 2(\delta + r_y) \;=\; 2\alpha(r_x + r_y) + \frac{2}{1+\beta}(r_x + r_y),$$

since $\delta = \alpha(r_x + r_y)$ and $1 + \beta = (r_x + r_y)/r_y$. Using diam $\mathcal{T}_{\text{di}} = (1+\alpha)(r_x + r_y)$ yields

$$\frac{\text{diam } \mathcal{T}_{\text{mono}}}{\text{diam } \mathcal{T}_{\text{di}}} \;\leq\; \frac{2\alpha(1+\beta) + 2}{(1+\alpha)(1+\beta)} \;=:\; f_{\text{mono}}(\alpha, \beta),$$

and the function $f_{\text{mono}}(\alpha, \beta)$ is an upper bound of $\mathcal{T}_{\text{mono}}$'s approximation factor.

In order to compute the maximum of the minimum of the two bounds we first analyze where $f_{\text{bisect}} \leq f_{\text{mono}}$. This is always the case if $\alpha \geq 2$ but also if $\alpha < 2$ and $\beta \leq g_{\text{equal}}(\alpha) := \frac{\alpha+2}{2-\alpha}$. See Figure 11 for the corresponding regions. Since neither f_{bisect} nor f_{mono} have any local or global maxima in the interior of the (α, β)-range we are interested in, we must consider their boundary values.

(1) For $\beta \equiv 1$ the tree $\mathcal{T}_{\text{bisect}}$ is optimal since $f_{\text{bisect}}(\alpha, 1) \equiv 1$.

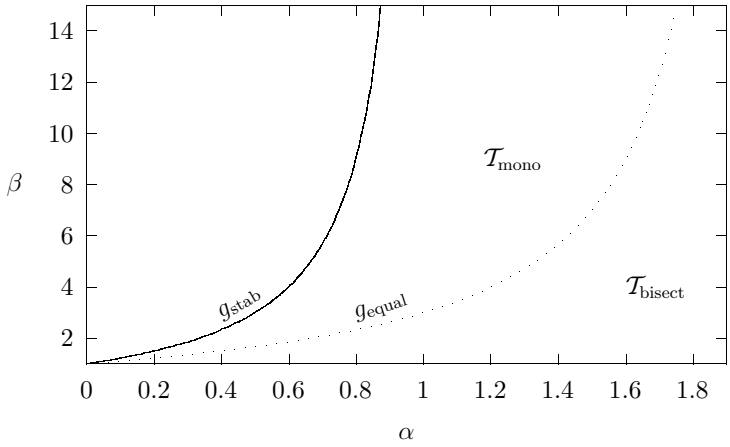

Fig. 11. Our upper bound for the approximation factor of $\mathcal{T}_{\text{mono}}$ ($\mathcal{T}_{\text{bisect}}$) is smaller to the left (right, respectively) of g_{equal}. To the left of g_{stab} the tree $\mathcal{T}_{\text{mono}}$ is optimal.

(2) Note that the stability lemma $r_x \leq \delta + r_y$ is equivalent to $\beta \leq g_{\text{stab}}(\alpha) := \frac{\alpha+1}{1-\alpha}$, see Figure 11. Along the graph of g_{stab} the tree $\mathcal{T}_{\text{mono}}$ is optimal since $f_{\text{mono}}(\alpha, g_{\text{stab}}(\alpha)) \equiv 1$.
(3) Along g_{equal} both functions equal $(3\alpha + 2)/(2\alpha + 2)$. This expression increases monotonically from 1 towards 4/3 when α goes from 0 towards 2.

Standard analysis of the partial derivatives shows that f_{mono} increases while f_{bisect} decreases monotonically when α goes to infinity. Thus the maximum of $\min(f_{\text{mono}}, f_{\text{bisect}})$ is indeed attained at g_{equal}. □

4 Approximation schemes for the MDST

In this section we give some fast approximation schemes for the MDST, i.e. factor-$(1+\varepsilon)$ approximation algorithms. The first approximation scheme uses a grid, the second and third use the well-separated pair decomposition, and the forth is a combination of the first and the third method. The reason for this multitude of approaches is that we want to take into account the way the running time depends not only on n, the size of the point set, but also on ε, the approximation factor.

Chan [Cha00] uses the following notation. Let $E = 1/\varepsilon$ and let the O^*-notation be a variant of the O-notation that hides terms of type $O(\log^{O(1)} E)$. (Such terms come into play e.g. when the use of the floor function is replaced by binary search with precision ε.) Then a *linear-time approximation scheme (LTAS) of order c* is a scheme with a running time of the form $O^*(E^c n)$ for

some constant c. A *strong LTAS of order c* has a running time of $O^*(E^c + n)$. Our asymptotically fastest scheme for approximating the MDST is a strong LTAS of order 5.

4.1 A grid-based approximation scheme

The idea of our first scheme is based on a grid which has been used before e.g. to approximate the diameter of a point set [BHP99,Cha00], i.e. the longest distance between any pair of the given points. We lay a grid of $O(E) \times O(E)$ cells over P, choose an arbitrary representative point for each cell and use the exact algorithm of Ho et al. [HLCW91] to compute the MDST $\mathcal{T}_R$ of the set R of all representative points. By connecting the remaining points in $P \setminus R$ to the pole adjacent to their representatives, we get a dipolar tree $\mathcal{T}_\varepsilon$ whose diameter is at most $(1+\varepsilon)$ times the tree diameter d_P of P.

The details are as follows. Let $M = \max_{p,q \in P}\{|x(p)x(q)|, |y(p)y(q)|\}$ be the edge length of the smallest enclosing square of P and let $l = \varepsilon M/(10\sqrt{2})$ be the edge length of the square grid cells. Clearly $M \leq d_P$. Since each path in T_ε is at most by two edges of length $l\sqrt{2}$ longer than the corresponding path in T_R we have diam $T_\varepsilon \leq$ diam $T_R + 2l\sqrt{2} \leq$ diam $T_R + \varepsilon d_P/5$. To see that diam $T_\varepsilon \leq (1+\varepsilon)\, d_P$ it remains to prove:

Lemma 7 diam $\mathcal{T}_R \leq (1 + 4\varepsilon/5)\, d_P$.

PROOF. Let $\mathcal{T}_P$ be a MDST of P that is either mono- or dipolar. Such a tree always exists according to [HLCW91].

Case I: $\mathcal{T}_P$ is monopolar. Let $x \in P$ be the pole of $\mathcal{T}_P$ and let $\rho_p \in R$ be the representative point of $p \in P$. Due to the definition of T_R we have

$$\text{diam } \mathcal{T}_R \leq \min_{x' \in R} \max_{s \neq t \in R} |sx'| + |x't| \leq \max_{s \neq t \in R} |s\rho_x| + |\rho_x t|.$$

(The first two terms are equal if there is a monopolar MDST of R, the last two terms are equal if there is a MDmST of R with pole ρ_x.) By triangle inequality

$$\text{diam } \mathcal{T}_R \leq \max_{s \neq t \in R} |sx| + |x\rho_x| + |\rho_x x| + |xt|,$$

i.e. we maximize the length of the polygonal chain (s, x, ρ_x, x, t) over all $s \neq t \in R$. By appending edges to points a and $b \in P$ in the grid cells of s and t, respectively, the length of the longest chain does not decrease, even if we now maximize over all $a, b \in P$ with $a \neq b$.

$$\text{diam } \mathcal{T}_R \leq \max_{a \neq b \in P} |a\rho_a| + |\rho_a x| + 2|x\rho_x| + |x\rho_b| + |\rho_b b|.$$

Using $|a\rho_a|, |x\rho_x|, |\rho_b b| \leq l\sqrt{2}$ and the triangle inequalities $|\rho_a x| \leq |\rho_a a| + |ax|$ and $|x\rho_b| \leq |xb| + |b\rho_b|$ yields diam $\mathcal{T}_R \leq 6l\sqrt{2} + \max_{a \neq b \in P} |ax| + |xb| = (1 + 3\varepsilon/5)d_P$.

Case II: $\mathcal{T}_P$ is dipolar. The analysis is very similar to case I, except the chains consist of more pieces. This yields diam $\mathcal{T}_R \leq 8l\sqrt{2} + \text{diam } \mathcal{T}_P = (1 + 4\varepsilon/5)\ d_P$. □

Theorem 2 *A spanning tree $\mathcal{T}_P$ of P with* diam $\mathcal{T}_P \leq (1 + 1/E) \cdot d_P$ *can be computed in $O^*(E^{6-1/3} + n)$ time using $O^*(E^2 + n)$ space.*

PROOF. In order to determine the grid cell of each point in P without the floor function, we do binary search—once on an x- and once on a y-interval of size M until we have reached a precision of l, i.e. we need $O(\log E)$ steps for each point. Using Chan's algorithm [Cha02] to compute T_R takes $\tilde{O}(|R|^{3-1/6})$ time and $\tilde{O}(|R|)$ space, where $|R| = O(E^2)$. □

4.2 The well-separated pair decomposition

Our second scheme uses the well-separated pair decomposition of Callahan and Kosaraju [CK95]. We briefly review this decomposition below.

Definition 1 *Let $\tau > 0$ be a real number, and let A and B be two finite sets of points in $\mathbb{R}^d$. We say that A and B are* well-separated w.r.t. τ*, if there are two disjoint d-dimensional balls C_A and C_B both of radius r such that $A \subset C_A$, $B \subset C_B$, and the distance between C_A and C_B is at least equal to τr.*

The parameter τ will be referred to as the *separation constant*. The following lemma follows easily from Definition 1.

Lemma 8 *Let A and B be two finite sets of points that are well-separated w.r.t. τ, let x and p be points of A, and let y and q be points of B. Then (i) $|xy| \leq (1 + 2/\tau) \cdot |xq|$, (ii) $|xy| \leq (1 + 4/\tau) \cdot |pq|$, (iii) $|px| \leq (2/\tau) \cdot |pq|$, and (iv) the angle between the line segments pq and py is at most* $\arcsin(2/\tau)$.

Definition 2 *Let P be a set of n points in $\mathbb{R}^d$, and $\tau > 0$ a real number. A* well-separated pair decomposition *(WSPD) for P (w.r.t. τ) is a sequence of pairs of non-empty subsets of P, $(A_1, B_1), (A_2, B_2), \ldots, (A_\ell, B_\ell)$, such that*

(1) A_i and B_i are well-separated w.r.t. τ for $i = 1, 2, \ldots, \ell$, and
(2) for any two distinct points p and q of P, there is exactly one pair (A_i, B_i) in the sequence such that (i) $p \in A_i$ and $q \in B_i$, or (ii) $q \in A_i$ and $p \in B_i$,

The integer ℓ is called the *size* of a WSPD. Callahan and Kosaraju show that a WSPD of size $\ell = O(\tau^2 n)$ can be computed using $O(n \log n + \tau^2 n)$ time and space. The WSPD will help us to limit our search for the two poles of an approximate MDdSTto a linear number of point pairs.

4.3 A straight-forward approximation scheme

The approximation algorithm consists of two subalgorithms: the first algorithm computes a MDmST and the second computes an approximation of the MDdST. We always output the one with smaller diameter. According to [HLCW91] there exists a MDST that is either a monopolar or a dipolar tree. The MDmST can be computed in time $O(n \log n)$, hence we will focus on the problem of computing a MDdST. Let $d_{\min}$ be the diameter of a MDdST and let $\mathcal{S}_{pq}$ denote a spanning tree with dipole $\{p, q\}$ whose diameter is minimum among all such trees. For any dipolar spanning tree $\mathcal{T}$ with dipole $\{u, v\}$ let $r_u(\mathcal{T})$ ($r_v(\mathcal{T})$) be the length of the longest edge of $\mathcal{T}$ incident to u (v, respectively) without taking into account the edge uv. When it is clear which tree $\mathcal{T}$ we refer to, we will use r_u and r_v.

Lemma 9 *Let $(A_1, B_1), \ldots, (A_\ell, B_\ell)$ be a WSPD of P w.r.t. τ, and let p and q be any two points in P. Then there is a pair (A_i, B_i) such that for every point $u \in A_i$ and every point $v \in B_i$ the inequality* $\operatorname{diam} \mathcal{S}_{uv} \leq (1 + 8/\tau) \cdot \operatorname{diam} \mathcal{S}_{pq}$ *holds.*

PROOF. According to Definition 2 there is a pair (A_i, B_i) in the WSPD such that $p \in A_i$ and $q \in B_i$. If u is any point in A_i and v is any point in B_i, then let $\mathcal{T}$ be the tree with poles u and v where u is connected to v, p and each neighbor of p in $\mathcal{S}_{pq}$ except q is connected to u, and q and each neighbor of q in $\mathcal{S}_{pq}$ except p is connected to v. By Lemma 8(ii) $|uv| \leq (1 + 4/\tau)|pq|$ and by Lemma 8(iii) $r_u \leq |up| + r_p \leq 2|pq|/\tau + r_p$. Since $\operatorname{diam} \mathcal{T} = r_u + |uv| + r_v$ we have

$$\operatorname{diam} \mathcal{T} \leq \left(r_p + 2\frac{|pq|}{\tau}\right) + \left(|pq| + 4\frac{|pq|}{\tau}\right) + \left(r_q + 2\frac{|pq|}{\tau}\right) < \left(1 + \frac{8}{\tau}\right) \operatorname{diam} \mathcal{S}_{pq}.$$

The lemma follows due to the minimality of $\mathcal{S}_{uv}$. □

A first algorithm is now obvious. For each of the $O(\tau^2 n)$ pairs (A_i, B_i) in a WSPD of P w.r.t. $\tau = 8E$ pick *any* point $p \in A_i$ and *any* point $q \in B_i$, sort P according to distance from p, and compute $\mathcal{S}_{pq}$ in linear time by checking every possible radius of a disk centered at p as in [HLCW91].

Lemma 10 *A dipolar tree $\mathcal{T}$ with* $\operatorname{diam} \mathcal{T} \leq (1+1/E) \cdot d_{\min}$ *can be computed in $O(E^2 n^2 \log n)$ time using $O(E^2 n + n \log n)$ space.*

Now we describe a more involved algorithm. It is asymptotically faster than the previous algorithm if $n = \Omega(E)$ (more precisely if $E = o(n \log n)$). We will prove its correctness in Section 4.5.

Theorem 3 *A dipolar tree $\mathcal{T}$ with* $\operatorname{diam} \mathcal{T} \leq (1+1/E) \cdot d_{\min}$ *can be computed in $O(E^3 n + En \log n)$ time using $O(E^2 n + n \log n)$ space.*

The idea of the algorithm is again to check only a linear number of pairs of points, using the WSPD, in order to speed up the computation of the disks around the two poles. Note that we need to find a close approximation of the diameters of the disks to be able to guarantee a $(1+\varepsilon)$-approximation of the MDdST. Obviously we cannot afford to try all possible disks for all possible pairs of poles. Instead of checking the disks we will show in the analysis that it suffices to check a constant number of ways to partition the input point set into two subsets, each corresponding to a pole. The partitions we consider are induced by a constant number of lines that are approximately orthogonal to the line through the poles. We cannot afford to do this for each possible pair. Instead we select a constant number of orientations and use a constant number of orthogonal cuts for each orientation. For each cut we calculate for each point in P the approximate distance to the farthest point on each side of the cut. Below we give a more detailed description of the algorithm. For its pseudocode refer to Algorithm 4.

Phase 1: Initializing. Choose an auxiliary positive constant $\kappa < \min\{0.9\varepsilon, 1/2\}$. As will be clear later, this parameter can be used to fine-tune which part of the algorithm contributes how much to the uncertainty and to the running time. In phase 3 the choice of the separation constant τ will depend on the value of κ and ε.

Definition 3 *A set of points P is said to be l-ordered if the points are ordered with respect to their orthogonal projection onto the line l.*

Let l_i be the line with angle $i\pi/\gamma$ to the horizontal line, where $\gamma = \lceil 4/\kappa \rceil$. This implies that for an arbitrary line l there exists a line l_i such that $\angle l_i l \leq \pi/(2\gamma)$. For $i = 1, \ldots, \gamma$, let F_i be a list of the input points sorted according to the l_i-ordering. The time to construct these lists is $O(\gamma n \log n)$.

For each l_i, rotate P and l_i such that l_i is horizontal. For simplicity we denote the points in P from left to right on l_i by $p_1, \ldots, p_n$. Let d_i denote the horizontal distance between p_1 and p_n. Let b_{ij}, $1 \leq j \leq \gamma$, be a marker on l_i at distance $jd_i/(\gamma+1)$ to the right of p_1. Let L_{ij} and R_{ij} be the set of points in P to the left and to the right of the vertical β_{ij} through b_{ij}, respectively.

Algorithm 4 Approx-MDdST(P, ε)

Phase 1: initializing
choose $\kappa \in (0, \min\{0.9\varepsilon, 1/2\})$; set $\gamma \leftarrow \lceil 4/\kappa \rceil$
for $i \leftarrow 1$ **to** γ **do**
  $l_i \leftarrow$ line with angle $i\pi/\gamma$ to the horizontal
  $F_i \leftarrow l_i$-ordering of P
end for i
for $i \leftarrow$ **to** γ **do**
  rotate P and l_i such that l_i is horizontal
  let $p_1, \ldots, p_n$ be the points in F_i from left to right
  $d_i \leftarrow |p_1.x - p_n.x|$
  for $j \leftarrow 1$ **to** γ **do**
    $b_{ij} \leftarrow$ marker on l_i at distance $jd_i/(\gamma + 1)$ to the right of p_1
    for $k \leftarrow 1$ **to** γ **do**
      $L'_{ijk} \leftarrow l_k$-ordered subset of F_k to the left of b_{ij}
      $R'_{ijk} \leftarrow l_k$-ordered subset of F_k to the right of b_{ij}
    end for k
  end for j
end for i
Phase 2: computing approximate farthest neighbors
for $i \leftarrow 1$ **to** γ **do**
  for $j \leftarrow 1$ **to** γ **do**
    for $k \leftarrow 1$ **to** n **do**
      $N(p_k, i, j, L) \leftarrow p_k$ {dummy}
      for $l \leftarrow 1$ **to** γ **do**
        $p_{\min} \leftarrow$ first point in L'_{ijl}; $p_{\max} \leftarrow$ last point in L'_{ijl}
        $N(p_k, i, j, L) \leftarrow$ point in $\{p_{\min}, p_{\max}, N(p_k, i, j, L)\}$ furthest from p_k
      end for l
    end for k
    repeat lines 20–26 with R instead of L
  end for j
end for i
Phase 3: testing pole candidates
$\tau = 8(\frac{1+\varepsilon}{(1+\varepsilon-(1+\kappa)(1+\kappa/24)} - 1)$
build WSPD for P with separation constant τ
$d \leftarrow \infty$ {smallest diameter so far}
for each pair (A, B) in WSPD **do**
  choose any two points $u \in A$ and $v \in B$
  find l_i with the smallest angle to the line through u and v
  $D \leftarrow \infty$ {approximate diameter of tree with poles u and v, ignoring $|uv|$}
  for $j \leftarrow 1$ **to** γ **do**
    $D \leftarrow \min\{D, |N(u, i, j, L)u| + |vN(v, i, j, R)|, |N(u, i, j, R)u| + |vN(v, i, j, L)|\}$
  end for j
  if $D + |uv| < d$ **then** $u' \leftarrow u$; $v' \leftarrow v$; $d \leftarrow D + |uv|$ **end if**
end for (A, B)
compute $\mathcal{T} \leftarrow \mathcal{S}_{u'v'}$
return $\mathcal{T}$

For each marker b_{ij} on l_i we construct γ pairs of lists, denoted L'_{ijk} and R'_{ijk}, where $1 \leq k \leq \gamma$. The list L'_{ijk} (R'_{ijk}) contains the points in L_{ij} (R_{ij}, respectively) sorted according to the l_k-ordering. Such a list can be constructed in $O(n)$ time since the ordering is given by F_k: we just have to filter out the points in F_k that are on the "wrong" side of β_{ij}. (Actually it is not necessary to store the whole lists L'_{ijk} and R'_{ijk}: we only need to store the first and the last point in each list.) Hence the total time complexity needed to construct the lists is $O(\gamma^3 n + \gamma n \log n)$, see lines 1–17 in Algorithm 4. These lists will help us to compute an approximate farthest neighbor in L_{ij} and R_{ij} for each point $p \in P$ in time $O(\gamma)$, as we describe below.

Phase 2: Computing approximate farthest neighbors. Let the approximate distance of a point q from p be the maximum distance among all projections of q onto the lines l_k. Now let the approximate farthest neighbor $N(p, i, j, L)$ of p be the point $q \in L_{ij}$ with maximum approximate distance from p. Each $N(p, i, j, L)$ can be computed in time $O(\gamma)$ by taking the farthest point from p over all first and last elements of L'_{ijk} with $k = 1, \ldots, \gamma$. Define and compute $N(p, i, j, R)$ analogously. Hence the total time complexity of phase 2 is $O(\gamma^3 n)$, as there are $O(\gamma^2 n)$ triples of type (p, i, j). The error we make by using approximate farthest neighbors is small:

Lemma 11 *If p is any point in P, p_L the point in L_{ij} farthest from p and p_R the point in R_{ij} farthest from p, then*

$$\textit{(a)} \quad |pp_L| \;\leq\; (1 + \kappa/24) \cdot |pN(p, i, j, L)| \quad \textit{and}$$

$$\textit{(b)} \quad |pp_R| \;\leq\; (1 + \kappa/24) \cdot |pN(p, i, j, R)|.$$

PROOF. Due to symmetry it suffices to check (a). If the algorithm did not select p_L as farthest neighbor it holds that for each of the l_k-orderings there is a point farther from p than p_L. Hence p_L must lie within a symmetric 2γ-gon whose edges are at distance $|pN(p, i, j, L)|$ from p. This implies that $|pp_L| \leq |pN(p, i, j, L)| / \cos(\pi/(2\gamma)) \leq |pN(p, i, j, L)| / \cos(\pi\kappa/8)$ using $\gamma = \lceil 4/\kappa \rceil$. Thus it remains to show that $1/\cos(\pi\kappa/8) \leq 1 + \kappa/24$. Since $\cos x \geq 1 - x^2/2$ for any x, the claim is true if $1 - \pi^2\kappa^2/128 \geq 1/(1 + \kappa/24)$. This inequality holds for all $0 < \kappa \leq 1/2$. $\square$

Phase 3: Testing pole candidates. Compute the WSPD for P with separation constant τ. To be able to guarantee a $(1+\varepsilon)$-approximation algorithm the value of τ will depend on ε and κ as follows:

$$\tau = 8 \left(\frac{1 + \varepsilon}{1 + \varepsilon - (1 + \kappa)(1 + \kappa/24)} - 1 \right).$$

Note that the above formula implies that there is a trade-off between the values τ and κ, which can be used to fine-tune which part of the algorithm contributes how much to the uncertainty and to the running time. Setting for instance κ to 0.9ε yields for ε small $16/\varepsilon + 15 < \tau/8 < 32/\varepsilon + 31$, i.e. $\tau = \Theta(1/\varepsilon)$. For each pair (A, B) in the decomposition we select two arbitrary points $u \in A$ and $v \in B$. Let $l_{(u,v)}$ be the line through u and v. Find the line l_i that minimizes the angle between l_i and $l_{(u,v)}$. That is, the line l_i is a close approximation of the direction of the line through u and v. From above we have that l_i is divided into $\gamma+1$ intervals of length $d_i/(\gamma+1)$. For each j, $1 \leq j \leq \gamma$, compute $\min(|N(u,i,j,L)u|+|vN(v,i,j,R)|, |N(u,i,j,R)u|+|vN(v,i,j,L)|)$. The smallest of these $O(\gamma)$ values is saved, and is a close approximation of diam $\mathcal{S}_{uv} - |uv|$, which will be shown below.

The number of pairs in the WSPD is $O(\tau^2 n)$, which implies that the total running time of the central loop of this phase (lines 33–41 in Algorithm 4) is $O(\gamma \cdot \tau^2 n)$. Building the WSPD and computing $\mathcal{S}_{u'v'}$ takes an extra $O(\tau^2 n + n \log n)$ time. Thus the whole algorithm runs in $O(\gamma^3 n + \gamma\tau^2 n + \gamma n \log n)$ time and uses $O(n \log n + \gamma^2 n + \tau^2 n)$ space. Setting $\kappa = 0.9\varepsilon$ yields $\gamma = O(E)$ and $\tau = O(E)$ and thus the time and space complexities we claimed.

4.5 The proof of correctness for Theorem 3

It remains to prove that the diameter of the dipolar tree that we compute is indeed at most $(1 + \varepsilon)\, d_{\min}$.

From Lemma 9 we know that we will test a pair of poles u and v for which diam $\mathcal{S}_{uv} \leq (1 + 8/\tau)\, d_{\min} = \frac{1+\varepsilon}{(1+\kappa)(1+\kappa/24)}\, d_{\min}$. The equality actually explains our choice of τ. In this section we will prove that our algorithm always computes a dipolar tree whose diameter is at most $(1+\kappa)(1+\kappa/24)$ diam $\mathcal{S}_{uv}$ and thus at most $(1 + \varepsilon)\, d_{\min}$.

Consider the tree $\mathcal{S}_{uv}$. For simplicity we rotate P such that the line l through u and v is horizontal and u lies to the left of v, as illustrated in Figure 12a. Let $\delta = |uv|$. Our aim is to prove that there exists an orthogonal cut that splits the point set P into two sets such that the tree obtained by connecting u to all points to the left of the cut and connecting v to all points to the right of the cut will give a tree whose diameter is a $(1+\kappa)$-approximation of diam $\mathcal{S}_{uv}$. Since the error introduced by approximating the farthest neighbor distances is not more than a factor of $(1+\kappa/24)$ according to Lemma 11, this will prove the claim in the previous paragraph.

Denote by C_u and C_κ the circles with center at u and with radius r_u and $r_\kappa = r_u + \kappa z$ respectively, where $z = \text{diam}\, \mathcal{S}_{uv} = \delta + r_u + r_v$. Denote by C_v the circle with center at v and with radius r_v. Let s and s' (t and t') be two points

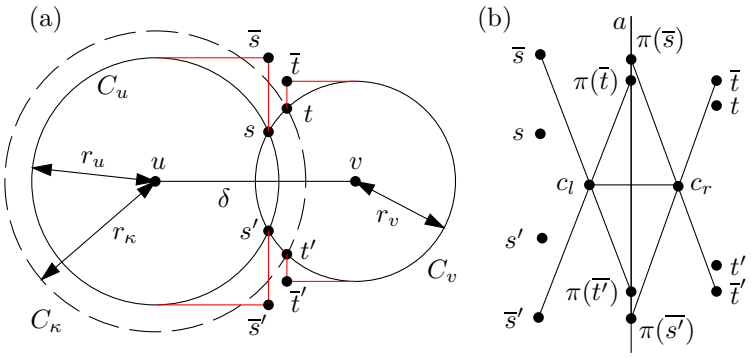

Fig. 12. A valid cut.

on C_u (C_v) such that if C_u (C_κ) and C_v intersect, then s and s' (t and t') are the two intersection points, where s (t) lies above s' (t', respectively). Otherwise, if C_u (C_κ) and C_v do not intersect, then $s = s'$ ($t = t'$) is the intersection of the line segment (u, v) and C_u (C_v, respectively), see Figure 12a.

We say that a cut with a line l_κ is *valid* iff all points in P to the left of l_κ are contained in C_κ and all points of P to the right of l_κ are contained in C_v. A valid cut guarantees a dipolar tree whose diameter is at most $\delta + r_\kappa + r_v = (1 + \kappa) \cdot \operatorname{diam} \mathcal{S}_{uv}$.

We will prove that the algorithm above always considers a valid cut. For simplicity we assume that $r_u(\mathcal{S}_{uv}) \geq r_v(\mathcal{S}_{uv})$. We will show that there always exists a marker b_{ij} on l_i such that cutting l_i orthogonally through b_{ij} is valid. Actually it is enough to show that the two requirements below are valid for any $\mathcal{S}_{uv}$. For a point p, denote the x-coordinate and the y-coordinate of p by $p.x$ and $p.y$, respectively. For simplicity we set $u = (0, 0)$. We have

$$\text{(i)} \quad \frac{z}{\gamma + 1} \cdot \frac{1}{\cos \frac{\pi}{2\gamma}} \leq \frac{1}{2}(t.x - s.x), \quad \text{and}$$

$$\text{(ii)} \quad \tan \frac{\pi}{2\gamma} \leq \frac{t.x - s.x}{2(r_u(\mathcal{S}_{uv}) + r_v(\mathcal{S}_{uv}))}.$$

The reason for this will now be explained. First we need to define some additional points. The reader is encouraged to study Figure 12 for a visual description. Let $\overline{s} = (s.x, r_u)$, $\overline{s}' = (s'.x, -r_u)$, $\overline{t} = (t.x, r_v)$ and $\overline{t}' = (t'.x, -r_v)$. Let a be the perpendicular bisector of the projections of s and t on the x-axis and let π be the orthogonal projection of the plane on a. Now we can define c_l to be the intersection point of the lines $(\overline{s}, \pi(\overline{t}'))$ and $(\overline{s}', \pi(\overline{t}))$, and c_r to be the intersection point of the lines $(\overline{t}, \pi(\overline{s}'))$ and $(\overline{t}', \pi(\overline{s}))$.

It now follows that any bisector l' that intersects the three line segments $(\overline{s}, \overline{t})$, (c_l, c_r) and $(\overline{s}', \overline{t}')$, will be a valid cut. This follows since all points to

the left of l' will be connected to u and all points to the right of l' will be connected to v, and the diameter of that tree will, obviously, be bounded by $\delta + (r_u(\mathcal{S}_{uv}) + \kappa z) + r_u(\mathcal{S}_{uv})$ which is a $(1+\kappa)$-approximation of diam $\mathcal{S}_{uv}$.

From the algorithm we know that (a) there is a line l_i such that $\angle(l_i, l_{(u,v)}) \leq \pi/(2\gamma)$, and that (b) there are γ orthogonal cuts of l_i that define equally many partitions of P. The distance between two adjacent orthogonal cuts of l_i is at most $z/(\gamma+1)$. This implies that the length of the largest interval on $l_{(u,v)}$ that is not intersected by any of these orthogonal cuts is at most

$$\frac{1}{\cos\frac{\pi}{2\gamma}} \cdot \frac{z}{\gamma+1}.$$

Hence requirement (i) ensures that for every $\mathcal{S}_{uv}$ the distance $|c_l c_r| = (t.x - s.x)/2$ must be large enough to guarantee that there is an orthogonal cut of l_i that intersects it.

An orthogonal cut of l_i has an angle of at least $\pi/2 - \pi/(2\gamma)$ to $l_{(u,v)}$. To ensure that an orthogonal cut of l_i that intersects the line segment $\overline{c_l c_r}$ also passes between $\overline{s}$ and $\overline{t}$ and between $\overline{s}'$ and $\overline{t}'$ it suffices to add requirement (ii).

It remains to prove the following lemma which implies that for every $\mathcal{S}_{uv}$ there is a valid orthogonal cut.

Lemma 12 *For any $u, v \in P$ $(u \neq v)$ the tree $\mathcal{S}_{uv}$ fulfills requirements (i) and (ii).*

PROOF. The tree $\mathcal{S}_{uv}$ can be characterized by the relationship of the two ratios

$$\alpha := \frac{\delta}{r_u + r_v} \quad \text{and} \quad F := \frac{1+\kappa/2}{1-\kappa/2}.$$

We distinguish three cases: (1) $\alpha < 1$, (2) $1 \leq \alpha \leq F$, and (3) $\alpha > F$. For each of these three cases we will show that $\mathcal{S}_{uv}$ fulfills the two requirements.

Case 1: Using the following two straight-forward equalities, $s.x^2 + s.y^2 = r_u^2$ and $(\delta - s.x)^2 + s.y^2 = r_v^2$, we obtain that $s.x = (\delta^2 + r_u^2 - r_v^2)/(2\delta)$. A similar calculation for $t.x$ yields $t.x = (\delta^2 + r_\kappa^2 - r_v^2)/(2\delta)$. Inserting these values gives $t.x - s.x = (\kappa^2 z^2 + 2\kappa z r_u)/(2\delta)$. The fact that $\alpha \leq F$ allows us to further simplify the expression for $t.x - s.x$ by using the following two expressions:

$$\frac{z}{\delta} = \frac{\delta + r_u + r_v}{\delta} = 1 + \frac{r_u + r_v}{\delta} \geq \frac{2}{1+\kappa/2}, \quad \text{and} \quad \frac{r_u}{\delta} \geq \frac{1-\kappa/2}{2(1+\kappa/2)}.$$

From this we obtain that

$$t.x - s.x = \frac{\kappa z}{2}\left(\frac{\kappa z}{\delta} + \frac{2r_u}{\delta}\right) > \frac{\kappa z}{2}.$$

This fulfills requirement (i) since

$$\frac{z}{\gamma+1}\cdot\frac{1}{\cos\frac{\pi}{2\gamma}} \;\leq\; \frac{\kappa z}{4} \;\leq\; \frac{1}{2}(t.x - s.x). \tag{1}$$

For requirement (ii) note that $\tan\pi/(2\gamma) \leq 2\kappa\tan\pi/16 < 2\kappa/5$. Since $\kappa \leq 1/2$ we get that $z/\delta \;\geq\; 2/(1+\kappa/2) \geq 8/5$. Combining this inequality, Equality 1, and our assumption that $r_u \;\geq\; r_v$ shows that requirement (ii) is also fulfilled:

$$\frac{t.x - s.x}{2(r_u+r_v)} \;\geq\; \frac{\kappa z}{4\delta}\left(\frac{2r_u+\kappa z}{r_u+r_v}\right) \;\geq\; \frac{\kappa z}{4\delta} \;\geq\; \frac{2\kappa}{5}.$$

Case 2: In this case we argue in the same manner as in the previous case. Using the fact that $s.x = r_u$ and $t.x = (\delta^2 + r_\kappa^2 - r_v^2)/(2\delta)$ yields

$$t.x - s.x \;\geq\; \frac{\kappa z}{2}\left(\frac{\kappa z}{\delta} + \frac{2r_u}{\delta}\right) > \frac{\kappa z}{2}.$$

The rest of the proof is exactly as in case 1.

Case 3: The first requirement is already shown to be fulfilled since $t.x - s.x \;\geq\; \delta - r_u - r_v \;\geq\; \kappa z/2$, hence it remains to show requirement (ii). We have

$$\frac{t.x - s.x}{2(r_u+r_v)} \;\geq\; \frac{\delta - (r_u+r_v)}{2(r_u+r_v)}$$

plugging in the values gives $\kappa/(2-\kappa)$, which is at least $2\kappa/5$. The lemma follows. □

The lemma says that for every dipole $\{u, v\}$ there exists a line a such that the dipolar tree obtained by connecting all the points on one side of a to u and all the points on the opposite side to v, is a $(1+\kappa)$-approximation of $\mathcal{S}_{uv}$.

4.6 Putting things together

Combining grid- and WSPD-based approach yields a strong LTAS of order 5:

Theorem 4 *A spanning tree $\mathcal{T}$ of P with* diam $\mathcal{T} \leq (1+1/E)\; d_P$ *can be computed in $O^*(E^5+n)$ time using $O(E^4+n)$ space.*

PROOF. Applying Algorithm 4 to the set $R \subseteq P$ of the $O(E^2)$ representative points takes $O(E^3|R| + E|R|\log|R|)$ time using $O(E^2|R| + |R|\log|R|)$ space according to Theorem 3. Connecting the points in $P \setminus R$ to the poles adjacent to their representative points yields a $(1+\varepsilon)$-approximation of the MDdST of P within the claimed time and space bounds as in Section 4.1. The difference

is that now the grid cells must be slightly smaller in order to compensate for the fact that we now approximate the MDdST of R rather than compute it exactly. A $(1+\varepsilon)$-approximation of the MDmST of P can be computed via the grid and an exact algorithm of Ho et al. [HLCW91] in $O^*(E^2+n)$ time using $O(E^2+n)$ space. Of the two trees the one with smaller diameter is a $(1+\varepsilon)$-approximation of the MDST of P. $\square$

Conclusions

On the one hand we have presented a new planar facility location problem, the discrete minimum-sum two-center problem that mediates between the discrete two-center problem and the minimum-diameter dipolar spanning tree. We have shown that there is an algorithm that computes the corresponding MSST in $O(n^2 \log n)$ time and that a variant of this tree is a factor-4/3 approximation of the MDST. It would be interesting to know whether there is a near quadratic-time algorithm for the MSST that uses $o(n^2)$ space.

On the other hand we have given four approximation schemes for the MDST. The asymptotically fastest is a combination of a grid-based approach with an algorithm that uses the well-separated pair decomposition. It computes in $O^*(\varepsilon^{-5}+n)$ time a tree whose diameter is at most $(1+\varepsilon)$ times that of a MDST. Such an algorithm is called a strong linear-time approximation scheme of order 5. Spriggs et al. [SKB$^+$03] recently improved our result by giving a strong LTAS of order 3 whose space consumption is linear in n and does not depend on ε. Is order 3 optimal? Is there an exact algorithm that is faster than Chan's [Cha02]? Is there a non-trivial lower bound on the computation time needed for the exact MDST?

Our scheme also works for higher-dimensional point sets, but the running time increases exponentially with the dimension. Linear-time approximation schemes for the discrete two-center problem and the MSST can be constructed similarly.

Acknowledgments

We thank an anonymous referee of an earlier version of this paper for suggesting Theorem 2. We also thank Pankaj K. Agarwal for pointing us to [Cha02] and Timothy Chan for sending us an updated version of [Cha02]. We thank another anonymous referee for pointing us to an error in the proof of Lemma 4. We finally thank Pat Morin for contributing the tree data structure in the proof of Lemma 4 that we needed to correct our error.

References

[AM95] Pankaj K. Agarwal and Jiří Matoušek. Dynamic half-space range reporting and its applications. *Algorithmica*, 13:325–345, 1995.

[ASW98] Pankaj K. Agarwal, Micha Sharir, and Emo Welzl. The discrete 2-center problem. *Discrete & Computational Geometry*, 20, 1998.

[BHP99] Gill Barequet and Sariel Har-Peled. Efficiently approximating the minimum-volume bounding box of a point set in three dimensions. In *Proc. 10th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'99)*, pages 82–91, Baltimore MA, 17–19 January 1999.

[Cha00] Timothy M. Chan. Approximating the diameter, width, smallest enclosing cylinder, and minimum-width annulus. In *Proc. 16th Annual Symposium on Computational Geometry (SoCG'00)*, pages 300–309, New York, 12–14 June 2000. ACM Press.

[Cha02] Timothy M. Chan. Semi-online maintenance of geometric optima and measures. In *Proc. 13th ACM-SIAM Symposium on Discrete Algorithms (SODA'02)*, pages 474–483, San Francisco, 6–8 January 2002.

[CK95] Paul B. Callahan and S. Rao Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *Journal of the ACM*, 42(1):67–90, January 1995.

[CLR90] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 1990.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, NY, 1979.

[HLCW91] Jan-Ming Ho, D. T. Lee, Chia-Hsiang Chang, and C. K. Wong. Minimum diameter spanning trees and related problems. *SIAM Journal on Computing*, 20(5):987–997, October 1991.

[HT95] Refael Hassin and Arie Tamir. On the minimum diameter spanning tree problem. *Information Processing Letters*, 53(2):109–111, 1995.

[SKB+03] Michael J. Spriggs, J. Mark Keil, Sergei Bespamyatnikh, Michael Segal, and Jack Snoeyink. Computing a $(1 + \epsilon)$-approximate geometric minimum-diameter spanning tree. *Algorithmica*, 2003. To appear.