Approximating the Geometric Minimum-Diameter Spanning Tree

Joachim Gudmundsson* Herman Haverkort† Sang-Min Park‡
Chan-Su Shin§ Alexander Wolff¶

March 16, 2002

Ernst-Moritz-Arndt-Universität Greifswald
Institut für Mathematik und Informatik
Preprint-Reihe Mathematik 4/2002

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 whose longest path consists of two or three edges, respectively. The more difficult dipolar case can so far only be solved in $O(n^3)$ time.

This paper has two aims. First, we present a solution to a new facility location problem, denoted 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 show that this problem can be solved in $O(n^2 \log n)$ time and that a variant of our solution yields a factor-1.2 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 $O(\frac{1}{\varepsilon^6} + n)$ time. The other uses a well-separated pair decomposition and takes $O(\frac{n}{\varepsilon^3} + \frac{n}{\varepsilon} \log n)$ time. A combination of the two approaches runs in $O(\frac{1}{\varepsilon^5} + n)$ time. Both can also be applied to MSST and 2CP.

Keywords

Computational geometry, algorithmic graph theory, geometric networks, approximation algorithms, linear-time approximation schemes.

*Dept. of Comp. Science, Utrecht University, The Netherlands. E-mail: joachim@cs.uu.nl
Supported by the Swedish Foundation for International Cooperation in Research and Higher Education.

†Dept. of Comp. Science, Utrecht University, The Netherlands. E-mail: herman@cs.uu.nl
Supported by the Netherlands' Organization for Scientific Research.

‡Dept. of Comp. Science, KAIST, Korea. E-mail: smpark@jupiter.kaist.ac.kr

§School of Electr. and Inform. Engineering, Hankuk University of Foreign Studies, Korea. E-mail: cssin@hufs.ac.kr

¶Institut für Mathematik und Informatik, Universität Greifswald, Germany. E-mail: awolff@uni-greifswald.de

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 that there always is a mono- or a dipolar MDST [10]. For a different proof, see [9]. Ho et al. gave an $O(n \log n)$-time algorithm for the monopolar and an $O(n^3)$-time algorithm for the dipolar case [10]. In the dipolar case the objective is to find the two roots $x, y \in P$ 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 that cover P.

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 largest disk 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 [8]. 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 by simply using the farthest-point Voronoi diagram of P. For $k = 2$, the problem becomes considerably harder. Using the notation from above, the 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. [2] 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 dipolar 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 cubic-time MDdST-algorithm and the subquadratic-time algorithm for the discrete two-center problem. More specifically, in Section 2 we will describe an algorithm that solves the discrete minimum-sum two-center problem in $O(n^2 \log n)$ time using $O(n^2)$ space, which is the first algorithm to address this problem.

In Section 3 we turn to our second aim, approximations for the MDST. To our knowledge nothing has been published on that field so far. We first give an $O(n \log n)$-time factor-$\frac{5}{3}$ approximation algorithm. Next we show that a variant of the MSST actually gives a factor-1.5 approximation for the MDST. Then we combine the MSST-variant with two other methods. We identify two parameters that depend on the MDdST and help to express a very tight estimation of the approximation factor of each of the three methods. The minimum over the three estimations turns out to be at most 1.2.

Finally, in Section 4 we show that there are even approximating schemes for the MDdST. More precisely, given a set P of n points and some $\varepsilon > 0$ we show how to compute a dipolar tree whose diameter is at most $(1 + \varepsilon)$ times as long as the diameter of a MDdST. Our first approximation scheme uses a grid of $\frac{1}{\varepsilon} \times \frac{1}{\varepsilon}$ square cells and runs the exact algorithm of Ho et al. [10] on one representative point per

cell. The same idea has been used before [3, 5] to approximate the diameter of a point set, i.e. the longest distance between any pair of the given points. Our scheme takes $O(\frac{1}{\varepsilon^6} + n)$ time if the floor function can be used in constant time, otherwise $O(\frac{1}{\varepsilon^6} + n \log \frac{1}{\varepsilon})$ time.

Our second approximation scheme is based on a well-separated pair decomposition [4] of P and takes $O(\frac{n}{\varepsilon^3} + \frac{n}{\varepsilon} \log n)$ time. Well-separated pair decompositions make it possible to consider only a linear number of pairs of points on the search for the two poles of an approximate MDdST. If we run our second scheme on the $O(\frac{1}{\varepsilon^2})$ representative points in the grid mentioned above, we get a new scheme with a running time of $O(\frac{1}{\varepsilon^5} + n)$ with the same restriction as above. Both schemes can also be applied to the MSST and the 2CP. We will refer to the diameter of the MDST of P as the *tree diameter* of P and we will abbreviate the *minimum-diameter monopolar spanning tree* by MDmST—analogously to the MDdST.

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. Then the MSST is the tree $\mathcal{T}_{\text{sum}}$ that minimizes what we will call the distance sum of $\mathcal{T}_{\text{sum}}$, namely $\text{ds}\, \mathcal{T}_{\text{sum}} = |pq| + \min\{|pm_{pq}|, |qm_{pq}|\}$. In order to give a faster algorithm for computing the MSST, we need a few definitions.

Let h_{pq} be the open halfplane that contains p and is delimited by the perpendicular bisector b_{pq} of p and q. Note that h_{pq}, h_{qp}, and b_{pq} partition the plane. Let f_{pq} be the point in $P \setminus h_{qp}$ that is farthest from p, see Figure 1. We call f_{pq} the *q-farthest point from p*. Let $|ab|$ be the Euclidean distance of the points a and b.

The first important idea of our algorithm is to split the problem of computing a MSST $\mathcal{T}_{pq}$ with dipole $\{p, q\}$ in two halves. We first compute f_{pq}, the q-farthest point from p. Actually we compute $f_{pq'}$ for all $q' \neq p$. This enables us to reuse information. Later we compute $f_{qp'}$ for all $p' \neq q$. Now the distance sum of $\mathcal{T}_{pq}$ is simply

$$\text{ds}\, \mathcal{T}_{pq} = |pq| + \max\{|pf_{pq}|, |qf_{qp}|\}. \tag{1}$$

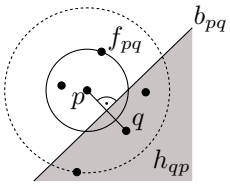

Figure 1: The q-farthest point from p, denoted by f_{pq}, is farthest from p among all points closer to p than to q.

Our algorithm consists of two phases. In phase I we go through all points p in P. The central (and time-critical) part of our algorithm is the computation of f_{pq} for all $q \in P \setminus \{p\}$. In phase II we then use Equation 1 to determine the spanning tree $\mathcal{T}_{pq}$ with dipole $\{p, q\}$ that has the smallest distance sum.

The second important observation that helped us to reduce the running time of the central part of our algorithm is the following. Let p be fixed. Instead of going through all $q \in P \setminus \{p\}$ and computing f_{pq} we characterize all q for which the q-farthest point f_{pq} of p is identical:

Lemma 1 *If $x \in P$ is the farthest point from $p \in P$, then x is the q-farthest point*

from p if and only if $q \notin D(x,p)$, where $D(a,b)$ (for points $a \neq b$) is the open disk that is centered at a and whose boundary contains b.

Proof. Since x is farthest from p, x is q-farthest from p if and only if $x \in h_{pq}$. This is the case iff the angle $\alpha = \angle pcx$ in the midpoint c of pq is at most 90 degrees, see Figure 2. Due to the Theorem of Thales this is equivalent to $c \notin D(m,p)$, where m is the midpoint of px. Finally this is equivalent to $q \notin D(x,p)$, since $d(p,q) = 2d(p,c)$ and $D(x,p)$ is the result of scaling $D(m,p)$ relative to p by a factor of 2. ☐

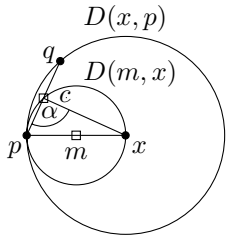

Figure 2: If x is farthest from p then x is q-farthest from p iff $q \notin D(x,p)$.

Using the above characterization we can label all points $q \in P \setminus \{p\}$ with the q-farthest point f_{pq} as follows. We first sort P in order of "decreasing" (i.e. non-increasing) distance from p. Let $q_1, q_2, \ldots, q_n = p$ be the resulting order. Label all points in the complement of $D(q_1,p)$ with q_1. Then label all *unlabeled* points in the complement of $D(q_2,p)$ with q_2. Continue this process until all points are labeled. Figure 3 visualizes the first three steps of this process. In that figure the light, medium, and dark shaded areas correspond to the areas in which all points are labeled with q_1, q_2, and q_3, respectively.

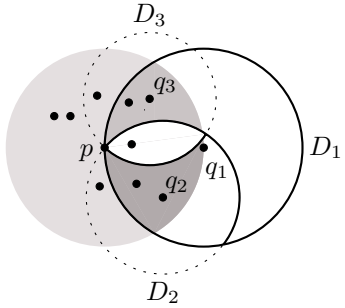

Figure 3: Labeling points with their q-farthest point.

It remains to show how all points $q \in P \setminus \{p\}$ can be labeled with f_{pq} efficiently. One approach would be to use dynamic circular range searching, which is equivalent to halfspace range searching in $\mathbb{R}^3$ [1]. The necessary data structure can be build in $O(n^{1+\varepsilon})$ time and space. After each query with a disk $D(q_i,p)$ all points that are *not* returned must be deleted. The total time for querying and updating is also $O(n^{1+\varepsilon})$. This would yield an $O(n^{2+\varepsilon})$-time algorithm. We will show that we can do better in the plane. However, it is not clear how our results can be generalized to higher dimensions. For dimensions 3 and 4 computing the MSST with range searching takes $O(n^{2.5} + \varepsilon)$ time [1] and thus is still faster than computing the MDST.

Lemma 2 *Given a set P of n points in the plane and given n disks $D_1, \ldots, D_n$ that all touch a point p, there is a data structure that allows to determine in $O(\log^2 n)$*

time for each point $q \in P$ the smallest integer i such that $q \in D_1 \cap \ldots \cap D_{i-1}$ and $q \notin D_i$ if such an integer exists. The data structure needs $O(n \log n)$ preprocessing time and space.

Proof. To simplify the presentation we assume $n = 2^k$. We build a complete binary tree $\mathcal{B}$ with k levels over n leaves with labels $1, \ldots, n$, see Figure 4. Each inner node v with left child l and right child r is labeled by a set of consecutive integers $\{a(v), \ldots, b(v)\} \subset \{1, \ldots, n\}$ that is recursively defined by $a(v) = a(l)$ and $b(v) = a(r) - 1$. For each leaf w we set $a(w) = b(w) = \text{label}(w)$. Note that the root is labeled $\{1, \ldots, n/2\}$. In Figure 4 $[a, b]$ is shorthand for $\{a, \ldots, b\}$.

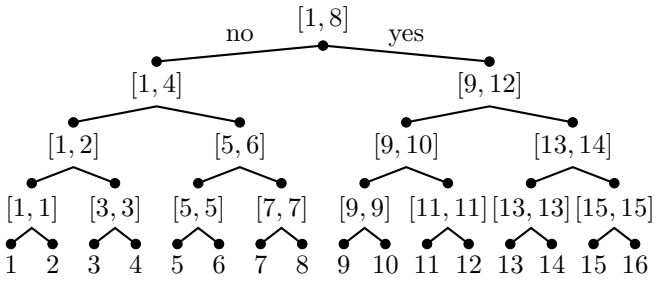

Figure 4: The binary search tree $\mathcal{B}$. A label $[a, b]$ of an inner node v indicates that in v for each query point q the test $q \in D_a \cap D_{a+1} \cap \ldots \cap D_b$ is performed.

A query with a point $q \in P$ consists of following a path from the root to a leaf whose label i is the index of the q-furthest point from p, in other words $p_i = f_{pq}$. In each inner node v the path of a query with point q is determined by testing whether $q \in D_{a(v)} \cap \ldots \cap D_{b(v)}$. If yes the next node of the query path is the right child, otherwise the left child. (Why such a query in deed gives the desired answer can be proven by induction over k.)

A query time of $O(\log^2 n)$ can be achieved by storing in each internal node v a decomposition of $D_{a(v)} \cap \ldots \cap D_{b(v)}$ into at most $b(v) - a(v) + 2$ vertical strips. The strips are bounded by all verticals through the endpoints of the arcs that for the boundary of $D_{a(v)} \cap \ldots \cap D_{b(v)}$. In this decomposition q can then be located in $O(\log n)$ time, and this has to be done $O(\log n)$ times on the way from the root to a leaf.

In order to construct the tree $\mathcal{B}$ we first build a tree $\mathcal{B}'$. The tree $\mathcal{B}'$ is also a binary tree over $\{1, \ldots, n\}$, but in $\mathcal{B}'$ each internal node v is labeled with the set of the labels of all leaves in the subtree rooted at v. The tree $\mathcal{B}'$ can be built in a bottom-up fashion since we can construct $D_1 \cap \ldots \cap D_{2m}$ from $D_1 \cap \ldots \cap D_m$ and $D_{m+1} \cap \ldots \cap D_{2m}$ by a merge-sort style procedure in $O(m)$ time. Note that each disk contributes at most one piece to the boundary of the intersection. Hence the construction of $\mathcal{B}'$ takes $O(n \log n)$ time (and space) in total.

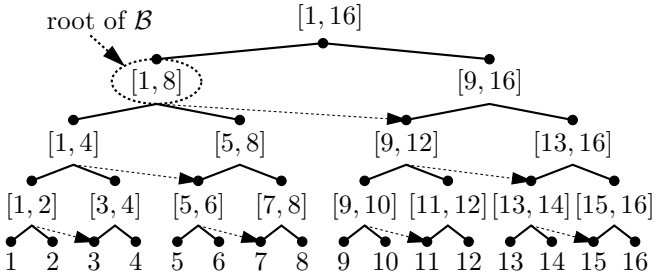

Figure 5: The auxiliary tree $\mathcal{B}'$ for the construction of $\mathcal{B}$ (see dotted arrows).

From $\mathcal{B}'$ we obtain $\mathcal{B}$ in three steps. First we make the left child of the root of $\mathcal{B}'$ the root of $\mathcal{B}$. Second for each node v we change the pointer from the right child of v to the left child of the sister of v, see the dotted arrows in Figure 5. Third, we make each leaf node w with label i in $\mathcal{B}'$ an internal node with label $\{i\}$ in $\mathcal{B}$ and add to w a left child with label i and a right child with label $i+1$ as new leaves. The vertical decomposition of each internal node v can then be computed in time linear in the size of the complexity of $D_{a(v)} \cap \ldots \cap D_{b(v)}$. Thus the construction of $\mathcal{B}$ takes $O(n \log n)$ time and space. ◻

The time complexity for querying the data structure can be reduced from $O(\log^2 n)$ to $O(\log n)$ by applying fractional-cascading techniques [6]. The time we need to reorganize the tree $\mathcal{B}$ to support fractional cascading is proportional to its space consumption, which is $O(n \log n)$ since there are $O(n)$ items on each level.

Theorem 1 *There is an algorithm that computes a minimum-sum dipolar spanning tree in $O(n^2 \log n)$ time using quadratic space.*

Proof. With the procedure described before Lemma 2 we can compute, for each $p \in P$, all points f_{pq} that are q-farthest from p in $O(n \log n)$ time using the data structure of Lemma 2 and fractional cascading. Thus we can compute all points of type f_{pq} in $O(n^2 \log n)$ time. Since we can only determine the tree with minimum distance sum *after* computing all points of type f_{pq} we must store them explicitly, which requires quadratic space. ◻

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 of P. In this section we will investigate various trees that give better approximations of the MDST at the expense of computation time and storage.

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 roots—have degree 1. The tree $\mathcal{T}_{\text{mono}}$ is a minimum-diameter star with vertex set P. Let x and y be the roots 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) without taking into account the edge xy. Wlog. we assume $r_x \geq r_y$.

3.1 A fast approximation

Let $\mathcal{T}_{\text{anti}}$ be the tree that is obtained by computing an antipodal pair $\{p, q\}$ of P and computing a MDdST with dipole $\{p, q\}$. The antipodal pair can be computed from the convex hull of P using Shamos' rotating-calipers algorithm [11]. Ho et al. [10] show how the minimum-diameter dipolar spanning tree of a fixed dipole $\{p, q\}$ can be found in linear time once all points have been sorted according to their distance to, say, p. Thus it takes $O(n \log n)$ time (and linear storage) to compute $\mathcal{T}_{\text{anti}}$.

The idea for our approximation is very simple. We consider two cases. If δ is small, then intuitively $\mathcal{T}_{\text{mono}}$ is a good approximation for $\mathcal{T}_{\text{di}}$, and if δ is large, then $\mathcal{T}_{\text{anti}}$ seems to approximate $\mathcal{T}_{\text{di}}$ well. Figures 6 and 7 depict nearly-worst case examples in the two cases. $\mathcal{T}_{\text{mono}}$ and $\mathcal{T}_{\text{anti}}$ are drawn with bold solid edges. Let $D_{c,r}$

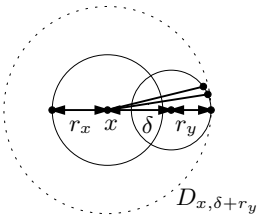

Figure 6: Approximating $\mathcal{T}_{\text{di}}$ with $\mathcal{T}_{\text{mono}}$.

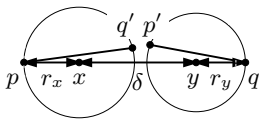

Figure 7: Approximating $\mathcal{T}_{\text{di}}$ with $\mathcal{T}_{\text{anti}}$.

be a disk with center c and radius r and let $\alpha = \frac{\delta}{r_x+r_y}$. Thus $\alpha > 0$, $\delta = \alpha(r_x + r_y)$ and diam $\mathcal{T}_{\text{di}} = (1+\alpha)(r_x + r_y)$.

Case I: δ is small.

The stability lemma $r_x < \delta + r_y$ [10] implies that $P \subseteq D_{x,\delta+r_y}$, see Figure 6. Due to that, the diameter of a monopolar spanning tree $\mathcal{T}$ rooted at x is at most twice the radius of $D_{x,\delta+r_y}$. We know that diam $\mathcal{T}_{\text{mono}} \leq$ diam $\mathcal{T}$ since $\mathcal{T}_{\text{mono}}$ is the MDmST of P. Thus diam $\mathcal{T}_{\text{mono}} \leq$ diam $\mathcal{T} \leq 2(\delta + r_y) \leq 2\delta + r_x + r_y = (1+2\alpha)(r_x + r_y)$ and

$$\frac{\text{diam } \mathcal{T}_{\text{mono}}}{\text{diam } \mathcal{T}_{\text{di}}} \leq \frac{1+2\alpha}{1+\alpha} = 2 - \frac{1}{1+\alpha}.$$

Case II: δ is large.

Due to our later choice of α we can assume that $p \in D_{x,r_x}$ and $q \in D_{y,r_y}$. Let $\mathcal{T}$ be a spanning tree with dipole $\{p, q\}$ where all points in $P \cap D_{x,r_x}$ are connected to p and all points in $P \cap D_{y,r_y}$ are connected to q. To bound the diameter of $\mathcal{T}$, observe that $|pq| \leq$ diam $\mathcal{T}_{\text{di}}$ and $|pq'| \leq 2r_x$ for all $q' \neq q$ that are connected to p. Similarly $|qp'| \leq 2r_y$ for all $p' \neq p$ that are connected to q, see Figure 7. This gives us the bound diam $\mathcal{T} \leq |pq| + \max|pq'| + \max|qp'| \leq \delta + 3(r_x + r_y)$. Since $\mathcal{T}_{\text{anti}}$ is the *minimum-diameter* dipolar spanning tree with dipole $\{p, q\}$, we know that

$$\frac{\text{diam } \mathcal{T}_{\text{anti}}}{\text{diam } \mathcal{T}_{\text{di}}} \leq \frac{\text{diam } \mathcal{T}}{\text{diam } \mathcal{T}_{\text{di}}} \leq \frac{3+\alpha}{1+\alpha} = 1 + \frac{2}{\alpha+1}.$$

If we set the approximation factors of $\mathcal{T}_{\text{mono}}$ and $\mathcal{T}_{\text{anti}}$ equal, we get $\alpha = 2$. For $\alpha \leq 2$ we have diam $\mathcal{T}_{\text{mono}} \leq \frac{5}{3}$ diam $\mathcal{T}_{\text{di}}$ due to case I. Accordingly in case II for the range $\alpha \geq 2$ we have diam $\mathcal{T}_{\text{anti}} \leq \frac{5}{3}$ diam $\mathcal{T}_{\text{di}}$. Thus computing $\mathcal{T}_{\text{anti}}$ and $\mathcal{T}_{\text{mono}}$ (in $O(n \log n)$ time each [11, 10]) and taking the one with smaller diameter gives us a factor-$\frac{5}{3}$ approximation of $\mathcal{T}_{\text{di}}$ for *all* values of α.

Lemma 3 *Given a set P of n points in the plane, there is a tree with the following two properties: it can be computed in $O(n \log n)$ time using linear storage, and its diameter is at most $\frac{5}{3}$ times as long as the tree diameter of P.*

3.2 A variant of the minimum-sum dipolar spanning tree

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 the tree $\mathcal{T}_{\text{sum}}$ of type $\mathcal{T}_{pq}$ with minimum *distance sum*, i.e. the tree that minimizes $|pq| + \max\{|f_{pq}p|, |qf_{qp}|\}$. Now we go through the 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 minimum-diameter dipolar spanning tree $\mathcal{T}_{\mathrm{di}}$.

As in the previous subsection we assume $r_x \geq r_y$. Now consider the tree $\mathcal{T}_{xy}$ and let r'_x (r'_y) be the length of the longest edge of $\mathcal{T}_{xy}$ incident to x (y) without taking into account the edge xy. Note that $r'_x = |xf_{xy}|$ and $r'_y = |yf_{yx}|$.

Lemma 4 $\operatorname{diam} \mathcal{T}_{\mathrm{bisect}} \leq \frac{3}{2} \operatorname{diam} \mathcal{T}_{\mathrm{di}}$.

Proof. We will compare the diameter of $\mathcal{T}_{xy}$ with that of $\mathcal{T}_{\mathrm{di}}$. First observe that $\max\{r'_x, r'_y\} \leq r_x$. This is due to the fact that f_{xy} and f_{yx} have at most distance r_x from both x and y. This observation yields $\operatorname{diam} \mathcal{T}_{xy} = r'_x + \delta + r'_y \leq 2\max\{r'_x, r'_y\} + \delta \leq 2r_x + \delta$. On the other hand $\operatorname{diam} \mathcal{T}_{\mathrm{di}} = r_x + \delta + r_y \geq 2r_x$ due to the stability lemma [10]. Combining these inequalities yields

$$\begin{aligned} \frac{\operatorname{diam} \mathcal{T}_{xy}}{\operatorname{diam} \mathcal{T}_{\mathrm{di}}} &\leq \frac{2r_x + \delta}{r_x + \delta + r_y} \leq \frac{(r_x + \delta + r_y) + r_x}{r_x + \delta + r_y} \\ &\leq 1 + \frac{r_x}{r_x + \delta + r_y} \leq 1 + \frac{r_x}{2r_x} = \frac{3}{2}. \end{aligned}$$

Observing that $\operatorname{diam} \mathcal{T}_{\mathrm{bisect}} \leq \operatorname{diam} \mathcal{T}_{xy}$ completes the proof. □

As in the previous subsection we must also compute $\mathcal{T}_{\mathrm{mono}}$ in order to treat the monopolar case which is not covered by our analysis. Choosing between $\mathcal{T}_{\mathrm{mono}}$ and $\mathcal{T}_{\mathrm{bisect}}$ the one with smaller diameter yields the following corollary of Lemma 4.

Corollary 1 *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 $\frac{3}{2}$ times as long as the tree diameter of P.*

3.3 A new tree

In this subsection we will construct a new tree $\mathcal{T}_{\mathrm{grow}}$. The analysis of its diameter only works if we assume $r_x < \delta$, i.e. if the edge xy between the two roots of $\mathcal{T}_{\mathrm{di}}$ is the longest edge of $\mathcal{T}_{\mathrm{di}}$. For point sets where this is not the case we will again take $\mathcal{T}_{\mathrm{mono}}$ to fill the gap. The algorithm for the construction of $\mathcal{T}_{\mathrm{grow}}$ is as follows.

For each point $p \in P$, go through the following steps.

1. Sort the points in P in order of non-decreasing distance from p. Let $p_1(=p), p_2, \ldots, p_n$ be the resulting sequence.
2. For $i = 2, \ldots, n-1$ let $A_i = \{p_1, \ldots, p_{i-1}\}$, $B_i = \{p_i, \ldots, p_n\}$ and $a_i = |pp_i|$.
3. For $i = 3, \ldots, n-1$ construct a spanning tree $\mathcal{S}_{p,i}$ with roots p and p_i. Connect the two roots and then the points in $A_i \setminus \{p\}$ to p and those in $B_i \setminus \{p_i\}$ to p_i. Among all trees $\mathcal{S}_{p,i}$, let $\mathcal{S}_p$ be one with minimum diameter.

Output the tree $\mathcal{T}_{\mathrm{grow}}$ with minimum diameter among the n trees of type $\mathcal{S}_p$.

See Figures 8 and 9 for how the sets of type A_i grow and how those of type B_i shrink during the computation of $\mathcal{S}_p$.

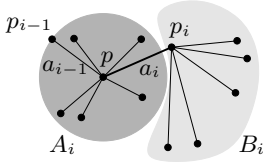

Figure 8: The tree $\mathcal{S}_{p,i}$.

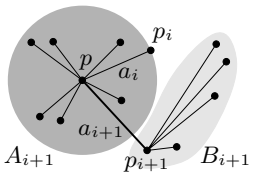

Figure 9: The tree $\mathcal{S}_{p,i+1}$.

Lemma 5 $\min\{\text{diam } \mathcal{T}_{\text{grow}}, \text{diam } \mathcal{T}_{\text{mono}}\} \leq \frac{4}{3}\text{diam } \mathcal{T}_{\text{di}}$.

Proof. In order to analyze the diameter of $\mathcal{T}_{\text{grow}}$ consider the tree $\mathcal{S}_{x,i}$ with roots x and p_i where i is chosen maximum such that $a_i = r_x$. The other root y of $\mathcal{T}_{\text{di}}$ is in B_i since we assumed $r_x < \delta$. Observe that *all* points of B_i (except y itself) are connected to y in $\mathcal{T}_{\text{di}}$ because they are farther from x than r_x. This means that the maximum distance (which is an upper bound for the distance of p_i to all other points in B_i) between any two points of B_i is at most $2r_y$. We also know that $a_i < \delta$ since $a_i = r_x$.

Thus we can bound the diameter of $\mathcal{S}_{x,i}$ as follows:

$$\text{diam } \mathcal{S}_{x,i} \;=\; a_{i-1} + a_i + \max_{j>i} |p_i p_j| \;\leq\; r_x + \delta + 2r_y$$

and $\text{diam } \mathcal{T}_{\text{grow}} \leq \text{diam } \mathcal{S}_x \leq \text{diam } \mathcal{S}_{x,i}$. For a finer analysis than in the previous subsections we define two constants α and β that only depend on $\mathcal{T}_{\text{di}}$: Let

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

hence $\alpha > 0$ and $\beta \geq 1$. Now

$$\frac{\text{diam } \mathcal{T}_{\text{grow}}}{\text{diam } \mathcal{T}_{\text{di}}} \;\leq\; 1 + \frac{r_y}{r_x + r_y + \delta} \;=\; 1 + \frac{r_y}{(1+\alpha)(r_x + r_y)}$$

since $\delta = \alpha(r_x + r_y)$. With $r_x = \beta r_y$ this simplifies to

$$\frac{\text{diam } \mathcal{T}_{\text{grow}}}{\text{diam } \mathcal{T}_{\text{di}}} \;\leq\; 1 + \frac{1}{(1+\alpha)(1+\beta)} \;=\; \frac{2 + \alpha + \beta + \alpha\beta}{(1+\alpha)(1+\beta)}.$$

Our assumption $r_x < \delta$ is equivalent to $\beta < g_{\text{grow-mono}}(\alpha) := \frac{\alpha}{1-\alpha}$; 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 10. To fill this gap we analyze the performance of $\mathcal{T}_{\text{mono}}$ more carefully depending on α and β. Using the inequality of case I in Subsection 3.1 we get

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

since $\delta = \alpha(r_x + r_y)$ and $1 + \beta = \frac{r_x + r_y}{r_y}$. Hence

$$\frac{\text{diam } \mathcal{T}_{\text{mono}}}{\text{diam } \mathcal{T}_{\text{di}}} \leq \frac{2(\alpha + \alpha\beta + 1)}{(1+\alpha)(1+\beta)}.$$

Let $f_{\text{grow}}(\alpha, \beta) = \frac{2+\alpha+\beta+\alpha\beta}{(1+\alpha)(1+\beta)}$ and $f_{\text{mono}}(\alpha, \beta) = \frac{2(\alpha+\alpha\beta+1)}{(1+\alpha)(1+\beta)}$ be the (upper bounds of the) approximation ratios that the trees $\mathcal{T}_{\text{grow}}$ and $\mathcal{T}_{\text{mono}}$ achieve, respectively. In order to compute the maximum of the minimum of f_{grow} and f_{mono} we first analyze where $f_{\text{grow}} \leq f_{\text{mono}}$. This is always the case if $\alpha \geq 1$ but interestingly enough also if $\alpha < 1$ and $\beta \leq g_{\text{grow-mono}}(\alpha)$, i.e. exactly for those values of α

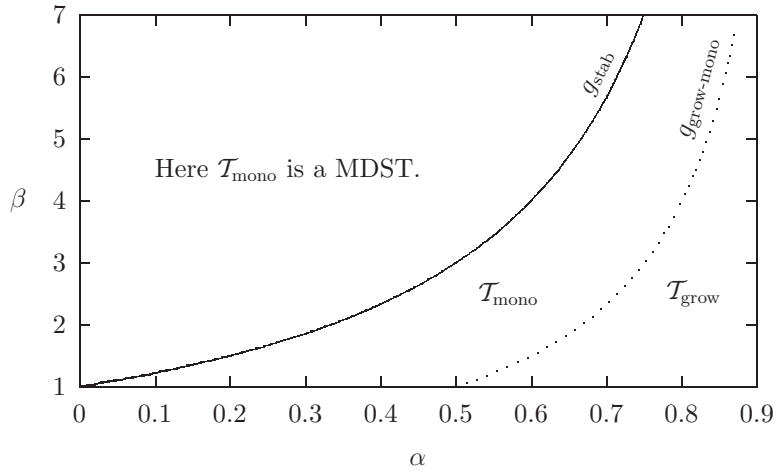

Figure 10: Our bound for $\mathcal{T}_{\text{mono}}$ is better above $g_{\text{grow-mono}}$; that for $\mathcal{T}_{\text{grow}}$ is better below. To the left of g_{stab} the tree $\mathcal{T}_{\text{mono}}$ has minimum diameter.

and β where the above analysis of the diameter of $\mathcal{T}_{\text{grow}}$ works. See Figure 10 for the corresponding regions. Since neither f_{grow} 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. Along g_{stab} we have that $f_{\text{mono}} \equiv 1$, along $g_{\text{grow-mono}}$ both functions $f_{\text{grow}}(\alpha) = f_{\text{mono}}(\alpha) = \frac{2}{1+\alpha}$ decrease monotonically from $\frac{4}{3}$ towards 1 when α goes from $\frac{1}{2}$ towards 1. For $\beta \equiv 1$ we have $f_{\text{mono}}(\alpha, 1) = 2 - \frac{1}{1+\alpha}$ which also decreases monotonically from $\frac{4}{3}$ to 1 when α goes from $\frac{1}{2}$ to 0, and we have $f_{\text{grow}}(\alpha, 1) = 1 + \frac{1}{2(1+\alpha)}$ which again decreases monotonically from $\frac{4}{3}$ towards 1 when α goes from $\frac{1}{2}$ to $+\infty$. □

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 $O(n^2 \log^2 n)$ time using $O(n \log n)$ storage, and its diameter is at most $\frac{4}{3}$ times as long as the tree diameter of P.*

Proof. It remains to prove the time and space bounds. In order to compute the diameter of $\mathcal{S}_{p,i}$ in step 3 of our algorithm we must maintain $\max_{j>i} |p_i p_j|$ efficiently for $i = 3, \ldots, n-1$. We do this by building for each $p \in P$ a binary tree with the same labels as the tree $\mathcal{B}'$ in Figure 5. In each inner node v with label $[a, b]$ we build a farthest-point Voronoi diagram of the points $p_a, p_{a+1}, \ldots, p_b$ including a data structure for point location in the diagram. This takes $O(n \log n)$ time and $O(n)$ space per level, thus $O(n \log^2 n)$ time and $O(n \log n)$ space for the whole tree.

After constructing this tree we query it once with each point p_i, $i = 3, \ldots, n-1$. A query consists of determining the point $p_j \in B_i$ (i.e. $j > i$) that is farthest from p_i. This can be done by following the canonical path from the root to the leaf with label i. Whenever we go to the left child of a node v on this path we locate p_i in $O(\log n)$ time in the farthest-point Voronoi diagram of the right child of v. This procedure gives us the points farthest from p_i in $O(\log n)$ sets whose union is B_i. In this way we can determine the point $p_j \in B_i$ farthest from p_i in $O(\log^2 n)$ time. Therefore step 3 of our algorithm takes $O(n \log^2 n)$ time for each point $p \in P$, and the whole algorithm takes $O(n^2 \log^2 n)$ time and needs $O(n \log n)$ space. □

Instead of maintaining $\max_{j>i} |p_i p_j|$ our analysis would also work if we could maintain the diameter of B_i for $i = 3, \ldots, n-1$. So far we were not able to do this faster than in $O(n \log^2 n)$ time [7]. Actually an approximation factor of $\frac{4}{3}$ can also

be achieved by combining $\mathcal{T}_{\text{bisect}}$ and $\mathcal{T}_{\text{mono}}$ and applying a similar "α-β-analysis". However, building $\mathcal{T}_{\text{bisect}}$ needs quadratic storage.

3.4 Putting things together

Now obviously it is interesting to see whether the combination of the results of the previous two subsections helps to approximate the tree diameter of P even better. It turns out that if we compute $\mathcal{T}_{\text{bisect}}$, $\mathcal{T}_{\text{grow}}$ and $\mathcal{T}_{\text{mono}}$ and take the one with minimum diameter, then we can in fact prove a better approximation factor:

Theorem 2 *Given a set P of n points in the plane there is a tree with the following two properties: it can be computed in $O(n^2 \log^2 n)$ time using $O(n^2)$ storage, and its diameter is at most $\frac{6}{5}$ times as long as the tree diameter of P.*

Proof. We apply our α-β-analysis to $\mathcal{T}_{\text{bisect}}$. Using $\delta = \alpha(r_x + r_y)$ and $r_x = \beta r_y$ the proof of Lemma 4 yields

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

since $2r_x = \frac{2\beta}{1+\beta}(r_x + r_y)$. Thus $f_{\text{bisect}}(\alpha, \beta) = \frac{\alpha+2\beta+\alpha\beta}{(1+\alpha)(1+\beta)}$ is an upper bound for the approximation factor that $\mathcal{T}_{\text{bisect}}$ achieves. This function does not have any extrema in the relevant section of the α-β-plane, so again we must investigate the boundaries of the regions where either f_{bisect}, f_{grow} or f_{mono} is minimum: (a) the inequality $f_{\text{bisect}} \leq f_{\text{grow}}$ is equivalent to $\beta \leq g_{\text{bisect-grow}}(\alpha) := 2$, (b) the inequality $f_{\text{bisect}} \leq f_{\text{mono}}$ is equivalent to $\alpha \geq 2$ or $\beta \leq g_{\text{bisect-mono}} := \frac{\alpha+2}{2-\alpha}$. See Figure 11 for the regions bounded by $g_{\text{bisect-mono}}$, $g_{\text{bisect-grow}}$ and $g_{\text{grow-mono}}$. Each region is labeled with the name of the tree whose bound is best in there. The maximum

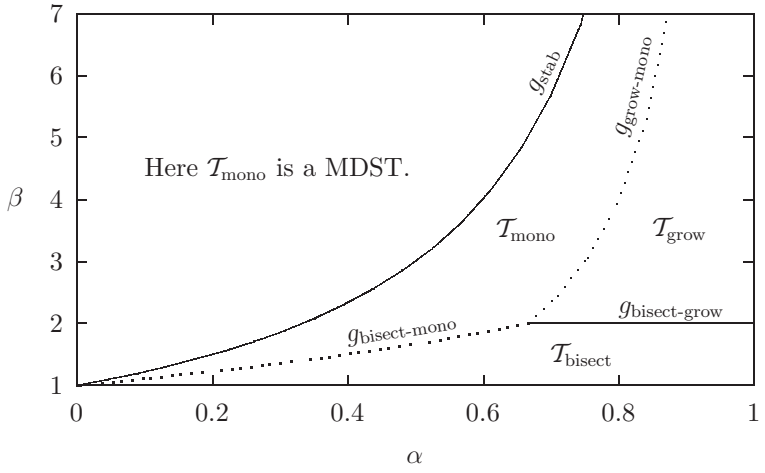

Figure 11: The part of the α-β-plane to the right of g_{stab} is split into three regions that are each labeled with the name of a tree. Within each region the corresponding bound on the tree's diameter is minimum.

of the minimum over the three functions f_{bisect}, f_{grow} and f_{mono} must occur on the boundary of the corresponding regions. Some basic calculus shows that the maximum is attained at the point $(\alpha, \beta) = (\frac{2}{3}, 2)$ where the boundaries of the corresponding regions meet. The value of the three functions in that point is $\frac{6}{5}$. □

4 Approximation schemes for the MDST

In this section we give a number of fast approximation schemes for the MDST, i.e. factor-$(1+\varepsilon)$ approximation algorithms. The first scheme uses a grid, the second and third a 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 [5] 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 floor function is replaced by binary search.) 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 best 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 [3, 5], i.e. the longest distance between any pair of the given points. We lay a grid of $E \times E$ cells over P, choose an arbitrary representative point for each cell and use the exact algorithm of Ho et al. [10] 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 their representative points we get a tree $\mathcal{T}_\varepsilon$ whose diameter is at most $(1+\varepsilon)\, d_P$, where d_P is the diameter of a MDST 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 $M \le d_P$. 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

$$\text{diam } T_\varepsilon \;\le\; \text{diam } T_R + 2l\sqrt{2} \;\le\; \text{diam } T_R + \frac{\varepsilon}{5} d_P.$$

To see that diam $T_\varepsilon \le (1+\varepsilon)\, d_P$ it remains to prove:

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

Proof. Let $\mathcal{T}_P$ be a MDST of P that is either mono- or dipolar. Such a tree always exists according to [10].

Case I: $\mathcal{T}_P$ is monopolar.

Let $x \in P$ be the root 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

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

(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 root ρ_x.) The triangle inquality now yields

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

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 $a \neq b$.

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

Using $|a\rho_a|, |x\rho_x|, |b\rho_b| \leq l\sqrt{2}$ and the triangle equality for $|\rho_a x| \leq |\rho_a a| + |ax|$ and similarily for $|\rho_b x|$ gives us

$$\text{diam } \mathcal{T}_R \;\leq\; 6l\sqrt{2} + \max_{a \neq b \in P} |ax| + |xb| \;=\; \frac{3}{5}\varepsilon + \text{diam } \mathcal{T}_P.$$

Case II: $\mathcal{T}_P$ is dipolar.

The analysis is very similar to case I, except the chains consist of more pieces, which gives the bound

$$\text{diam } \mathcal{T}_R \;\leq\; 8l\sqrt{2} + \text{diam } \mathcal{T}_P \;=\; (1 + \frac{4}{5}\varepsilon)\, d_P.$$

□

Theorem 3 *A spanning tree $\mathcal{T}_P$ of P with* diam $\mathcal{T}_P \leq (1+1/E)\cdot d_P$ *can be computed in $O^*(E^6 + n)$ 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 twice binary search on an interval of size M until we have reached a precision of l, i.e. we need $O(\log E)$ steps for each point. Using the algorithm of Ho et al. [10] to compute T_R takes $O(|R|^3)$ time and uses $O(|R|)$ space where $|R| \leq E^2$. □

4.2 The well-separated pair decomposition

Our second scheme uses the well-separated pair decomposition of Callahan and Kosaraju [4]. 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 the 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.

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 [10] there exists a MDST that is either a monopolar or a dipolar tree. The minimum-diameter monopolar tree 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 $\mathcal{T}_{pq}$ 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) without taking into account the edge uv. When it is clear which tree we refer to, we will use r_u and r_v.

Observation 1 *Let $(A_1, B_1), \ldots, (A_\ell, B_\ell)$ be a well-separated pair decomposition (WSPD) of P with separation constant 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* diam $\mathcal{T}_{uv} \leq (1 + 8/\tau) \cdot$ diam $\mathcal{T}_{pq}$ *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 $\mathcal{T}_{pq}$ except q is connected to u, and q and each neighbor of q in $\mathcal{T}_{pq}$ 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 diam $\mathcal{T} = r_u + |uv| + r_v$ we have

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

The lemma follows since $\mathcal{T}_{uv}$ has minimum diameter among all dipolar spanning trees with dipole $\{u, v\}$. ◻

A first algorithm is now obvious. For each pair (A_i, B_i) in the WSPD of P pick *any* vertex $p \in A_i$ and *any* vertex $q \in B_i$, and compute $\mathcal{T}_{pq}$. This is done by checking every possible radius of a disk centered at p as in [10] in $O(n \log n)$ time. We summarize:

Lemma 9 *A dipolar tree $\mathcal{T}$ with* diam $\mathcal{T} \leq (1 + 1/E)\, d_{\min}$ *can be computed in $O(E^2 n^2 \log n)$ time using $O(E^2 n + n \log n)$ space.*

Proof. Setting the separation constant τ to $8/\varepsilon$, the lemma follows from Observation 1 and from the fact that all possible radii are tested. The running time is dominated by sorting P for each of the $O(\tau^2 n)$ pairs in the WSPD, while the space consumption is dominated by that of the WSPD. ◻

4.4 A fast approximation scheme

Now we describe a faster algorithm and show its time complexity; we will prove its correctness in Section 4.5.

Theorem 4 *A dipolar tree $\mathcal{T}$ with* 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 a linear number of pairs of points, using the WSPD, but 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 partitions of the points among the poles. The partition of points is done by cuts that are 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 1.

Algorithm 1: Approx-MDdST(P, ε)

Ensure: diam $\mathcal{T} \leq (1+\varepsilon)\, d_{\min}$
Phase 1: initializing
1: choose $\kappa \in (0, \min\{0.9\varepsilon, 1/2\})$
2: $\gamma \leftarrow \lceil 4/\kappa \rceil$
3: **for** $i \leftarrow 1$ **to** γ **do**
4: $\quad l_i \leftarrow$ line with angle $i\frac{\pi}{\gamma}$ to the horizontal
5: $\quad F_i \leftarrow l_i$-ordering of P
6: **end for**
7: **for** $i \leftarrow$ **to** γ **do**
8: $\quad$ rotate P and l_i such that l_i is horizontal
9: $\quad$ let $p_1, \ldots, p_n$ be the points in F_i from left to right
10: $\quad d_i \leftarrow |p_1.x - p_n.x|$
11: $\quad$ **for** $j \leftarrow 1$ **to** γ **do**
12: $\quad\quad b_{ij} \leftarrow$ point on l_i at dist. $j\frac{d_i}{\gamma+1}$ to the right of p_1
13: $\quad\quad$ **for** $k \leftarrow 1$ **to** γ **do**
14: $\quad\quad\quad L'_{ijk} \leftarrow l_k$-ordered subset of F_k to the left of b_{ij}
15: $\quad\quad\quad R'_{ijk} \leftarrow l_k$-ordered subset of F_k to the right of b_{ij}
16: $\quad\quad$ **end for**
17: $\quad$ **end for**
18: **end for**
Phase 2: computing approximate farthest neighbors
19: **for** $i \leftarrow 1$ **to** γ **do**
20: $\quad$ **for** $j \leftarrow 1$ **to** γ **do**
21: $\quad\quad$ **for** $k \leftarrow 1$ **to** n **do**
22: $\quad\quad\quad N(p_k, i, j, L) \leftarrow p_k$ (dummy)
23: $\quad\quad\quad$ **for** $l \leftarrow 1$ **to** γ **do**
24: $\quad\quad\quad\quad p_{\min} \leftarrow$ first point in L'_{ijl}
25: $\quad\quad\quad\quad p_{\max} \leftarrow$ last point in L'_{ijl}
26: $\quad\quad\quad\quad$ **if** $|p_k p_{\min}| > |p_k p_{\max}|$ **then**
27: $\quad\quad\quad\quad\quad f \leftarrow p_{\min}$
28: $\quad\quad\quad\quad$ **else**
29: $\quad\quad\quad\quad\quad f \leftarrow p_{\max}$
30: $\quad\quad\quad\quad$ **end if**
31: $\quad\quad\quad\quad$ **if** $|p_k f| > |p_k N(p_k, i, j, L)|$ **then**
32: $\quad\quad\quad\quad\quad N(p_k, i, j, L) \leftarrow f$
33: $\quad\quad\quad\quad$ **end if**
34: $\quad\quad\quad$ **end for**
35: $\quad\quad$ **end for**
36: $\quad\quad$ repeat lines 21–35 with R instead of L
37: $\quad$ **end for**
38: **end for**

Phase 3: testing pole candidates

39: $\tau = 8(\frac{1+\varepsilon}{(1+\varepsilon-(1+\kappa)(1+\kappa/24)} - 1)$
40: build WSPD for P with separation constant τ
41: $d \leftarrow \infty$ {smallest diameter so far}
42: **for** each pair (A, B) in WSPD **do**
43: choose any two points $u \in A$ and $v \in B$
44: $D \leftarrow \infty$ {approx. diam. of tree with poles u and v}
45: $l_{(u,v)} \leftarrow$ the line through u and v
46: find l_i such that $\angle l_i l_{(u,v)}$ is minimized
47: **for** $j \leftarrow 1$ **to** γ **do**
48: $D \leftarrow \min\{D,$
$|N(u,i,j,L)u| + |uv| + |vN(v,i,j,R)|,$
$|N(u,i,j,R)u| + |uv| + |vN(v,i,j,L)|\}$
49: **end for**
50: **if** $D < d$ **then**
51: $u' \leftarrow u$ and $v' \leftarrow v$
52: $d \leftarrow D$
53: **end if**
54: **end for**
55: compute $\mathcal{T} \leftarrow \mathcal{T}_{u'v'}$
56: **return** $\mathcal{T}$

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 $\frac{i\pi}{\gamma}$ 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 $\angle l_i l \leq \frac{\pi}{2\gamma}$. For each i, $1 \leq i \leq \gamma$, sort the input points with respect to the l_i-ordering. We obtain γ sorted lists $F = \{F_1, \ldots, F_\gamma\}$. Each point p in F_i has a pointer to itself in $F_{(i \bmod \gamma)+1}$. 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 and consider the orthogonal projection of the points in P onto l_i. 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 the point on l_i at distance $\frac{jd_i}{\gamma+1}$ to the right of p_1. Let L_{ij} and R_{ij} be the set of points to the left and to the right of b_{ij} respectively.

For each point b_{ij} on l_i we construct γ pairs of lists, denoted L'_{ijk} and R'_{ijk}, where $1 \leq k \leq \gamma$. A list L'_{ijk} (R'_{ijk}) contains the set of points in L_{ij} (R_{ij}) sorted according to the l_k-ordering. Such a list can be constructed in linear time since the ordering is given by the list F_k. (Actually it is not necessary to store the 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 1–18 in Algorithm 1. 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 will be described below.

Phase 2: Computing approximate farthest neighbors. Compute, for each point p, an approximate farthest neighbor in L_{ij} and an approximate farthest neighbor in R_{ij}, denoted $N(p,i,j,L)$ and $N(p,i,j,R)$ respectively. This can be done in

time $O(\gamma)$ by using the lists L'_{ijk} and R'_{ijk}: just compute the distance between p and the first respectively the last point in each list. There are γ lists for each pair (i, j) and given that at most two entries in each list have to be checked, an approximate farthest neighbor can be computed in time $O(\gamma)$. Hence the total time complexity of this phase 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:

Observation 2 *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 (a) $|pp_L| \leq (1 + \kappa/24) \cdot |pN(p, i, j, L)|$
and (b) $|pp_R| \leq (1 + \kappa/24) \cdot |pN(p, i, j, R)|$.

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_i-orderings there is a point further 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 $|pN(p, i, j, L)| \geq |pp_L| \cos(\pi/(2\gamma)) \geq |pp_L|/(1 + \kappa/24)$, using some basic calculus and $\kappa \leq 1/2$. □

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. $\tau = \Theta(\frac{1}{\varepsilon})$. 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 $\frac{d_i}{\gamma+1}$. For each value of j, $1 \leq j \leq \gamma$, compute $\min(|N(u, i, j, L)u| + |uv| + |vN(v, i, j, R)|, |N(u, i, j, R)u| + |uv| + |vN(v, i, j, L)|)$. The smallest of these $O(\gamma)$ values is saved, and is a close approximation of the diameter of $\mathcal{T}_{uv}$, 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 42–54 in Algorithm 1) is $O(\gamma \cdot \tau^2 n)$. Building the WSPD and computing $\mathcal{T}_{u'v'}$ 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 $\gamma = O(\frac{1}{\varepsilon})$ and $\tau = O(\frac{1}{\varepsilon})$ and thus the time and space complexities we claimed. It remains to prove that the diameter of the dipolar tree that we compute is indeed at most $(1 + \varepsilon) \cdot d_{\min}$.

4.5 The proof of correctness

From Observation 1 we know that we will test a pair of poles u and v for which diam $\mathcal{T}_{uv} \leq (1 + 8/\tau)\ d_{\min} = \frac{1+\varepsilon}{(1+\kappa)(1+\kappa/24)}\ d_{\min}$. 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 $\mathcal{T}_{uv}$ and thus at most $(1 + \varepsilon)\ d_{\min}$.

Consider the tree $\mathcal{T}_{uv}$. 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

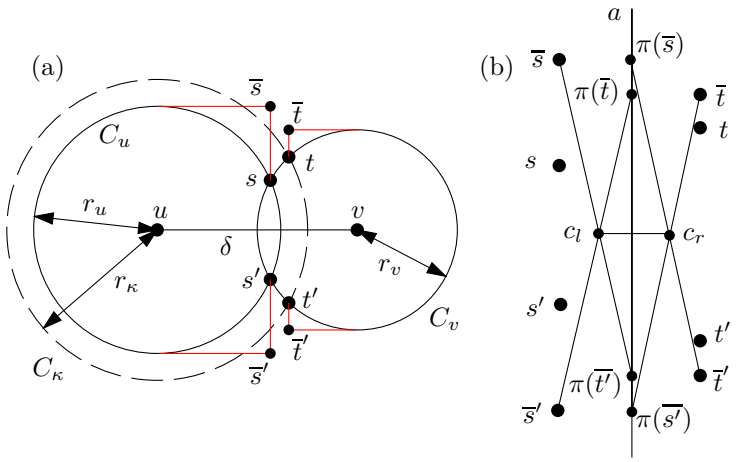

Figure 12: A valid cut.

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 $\mathcal{T}_{uv}$. Since the error introduced by approximating the farthest neighbor distances is not more than a factor of $(1+\kappa/24)$ according to Observation 2, 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 $z = \text{diam } \mathcal{T}_{uv} = \delta + r_u + r_v$. 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 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'). 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), 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 $\delta + r_\kappa + r_v = (1+\kappa) \cdot \text{diam } \mathcal{T}_{uv}$.

We will prove that the algorithm above always considers a valid cut. For simplicity we assume that $r_u(\mathcal{T}_{uv}) \geq r_v(\mathcal{T}_{uv})$. We will show that there always exists a point 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 $\mathcal{T}_{uv}$. 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 $u = (0, 0)$.

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

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

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 $\delta+(r_u(\mathcal{T}_{uv})+\kappa z)+r_u(\mathcal{T}_{uv})$ which is a $(1+\kappa)$-approximation of diam $\mathcal{T}_{uv}$.

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 $\mathcal{T}_{uv}$ 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 $\mathcal{T}_{uv}$ there is a valid orthogonal cut.

Lemma 10 *For any $u, v \in P$ $(u \neq v)$ the tree $\mathcal{T}_{uv}$ fulfills the requirements (i) and (ii).*

Proof. The tree $\mathcal{T}_{uv}$ 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 $\mathcal{T}_{uv}$ 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:

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

$$\frac{r_u}{\delta} \geq \frac{1-\kappa/2}{2(1+\kappa/2)}.$$

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

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

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 2, 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 $\mathcal{T}_{uv}$.

4.6 Putting things together

By combining the grid-based approach of Section 4.1 with the fast approximation scheme of Section 4.4 we get a LTAS of order 5:

Theorem 5 *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 1 to the set $R \subseteq P$ of at most 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 4. Connecting the points in $P \setminus R$ to their representative points yields a $(1+\varepsilon)$-approximation of the MDdST of P within the claimed time and space bounds in a similar way as in Section 4.1. The only difference is that here 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. [10] in $O^*(E^2 + n)$ time using $O(E^2 + n)$ space. The tree with the smaller diameter of the two approximations is a $(1+\varepsilon)$-approximation of the MDST of P. □

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-1.2 approximation for the MDST.

We would like to find a way to reduce the space consumption of our algorithm to $o(n^2)$.

On the other hand we have given a number of fast approximation schemes for the MDST. The fastest is a combination of a grid-based approach with an algorithm that uses a well-seperated pair decomposition. It computes in $O(1/\varepsilon^5 + n)$ time a tree whose diameter is at most $(1 + \varepsilon)$ times that of a MDST. Such a scheme is called a linear-time approximation scheme of order 5. The scheme also works for higher dimensional point sets, but the running time increases exponentially with the dimension. There are similar linear-time approximation schemes for the discrete two-center problem and the MSST.

References

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

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

[3] 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.

[4] 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.

[5] 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.

[6] Bernard Chazelle and Leonidas J. Guibas. Fractional cascading: I. A data structuring technique. *Algorithmica*, 1(3):133–162, 1986.

[7] David P. Dobkin and Subhash Suri. Maintenance of geometric extrema. *J. ACM*, 38:275–298, 1991.

[8] 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.

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

[10] 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.

[11] Franco P. Preparata and Michael Ian Shamos. *Computational Geometry An Introduction*. Springer-Verlag, New York, 1985.