Facility Location and the Geometric Minimum-Diameter Spanning Tree*

Joachim Gudmundsson[1], Herman Haverkort[2], Sang-Min Park[3], Chan-Su Shin[4], and Alexander Wolff[5]

[1] Dept. of Comp. Science, Utrecht University, The Netherlands. joachim@cs.uu.nl
[2] Dept. of Comp. Science, Utrecht University, The Netherlands. herman@cs.uu.nl
[3] Dept. of Comp. Science, KAIST, Korea. smpark@jupiter.kaist.ac.kr
[4] School of Electr. and Inform. Engineering, Hankuk University of Foreign Studies, Korea. cssin@hufs.ac.kr
[5] Institut für Mathematik und Informatik, Universität Greifswald, Germany. awolff@uni-greifswald.de

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 show that this problem can be solved in $O(n^2 \log n)$ time and that it yields a factor-4/3 approximation of the MDST.

Second, we give two fast approximation schemes for the MDST. 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.

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

* J. Gudmundsson was supported by the Swedish Foundation for International Cooperation in Research and Higher Education, H. Haverkort by the Netherlands' Organization for Scientific Research, C.-S. Shin by grant No. R05-2002-000-00780-0 from the Basic Research Program of the Korea Science and Engineering Foundation (KOSEF), and A. Wolff by the KOSEF program Brain Korea 21.

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]. In the same paper they also gave an $O(n \log n)$-time algorithm for the monopolar and an $O(n^3)$-time algorithm for the dipolar case. The latter time bound was recently improved by Chan [6] 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 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 [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 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 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 [6] that can easily be modified to compute the MSST instead.

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.[6] 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 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 $O(E) \times O(E)$ square cells (where $E = 1/\varepsilon$) and runs Chan's exact algorithm [6] 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 $O^*(E^6 + n)$ time, where the O^*-notation hides terms of type $O(\log^{O(1)} E)$.

Our second approximation scheme is based on the well-separated pair decomposition [4] of P and takes $O(E^3 n + En \log n)$ 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 $O(E^2)$ representative points in the grid mentioned above, we get a new scheme with a running time of $O^*(E^5 + n)$. Both schemes can also be applied to the MSST and the 2CP.

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 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 $\mathcal{T}_{pq}$ 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.) Clearly the tree $\mathcal{T}_{pq}$ that minimizes $|pq| + \min\{|pm_{pq}|, |qm_{pq}|\}$ is a MSST.

The first important idea of our algorithm is to 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) the point $f_{pq} \in P \setminus h_{qp}$ that is farthest from p ignoring the halfplane of q, see Figure 1. We call f_{pq} the *q-farthest point from p*. Now we want to find the tree $\mathcal{T}_{pq}$ that minimizes $|pq| + \max\{|pf_{pq}|, |qf_{qp}|\}$. This strategy enables us to reuse information.

[6] Very recently we were informed of [11] where the authors give an approximation scheme for the MDST that runs in $O(\varepsilon^{-3} + n)$ time using $O(n)$ space.

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 the above form of our target function to determine the MSST.

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. □

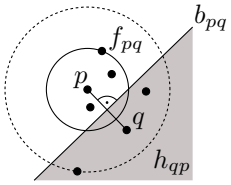

Fig. 1. The q-farthest point f_{pq} from p is farthest from p among all points closer to p than to q.

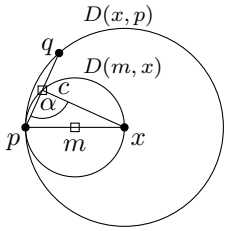

Fig. 2. If x is farthest from p then x is q-farthest from p iff $q \notin D(x,p)$.

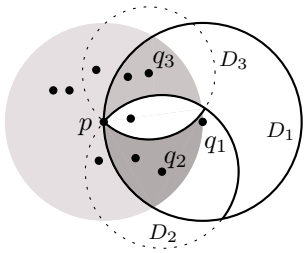

Fig. 3. Labeling points with their q-farthest point.

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 areas shaded light, medium, and dark correspond to the areas in which all points are labeled with q_1, q_2, and q_3, respectively.

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 $d \in \{3, 4\}$ computing the MSST with range searching takes $O(n^{2.5+\varepsilon})$ time [1] and thus is still faster than Chan's $\tilde{O}(n^{3-c_d})$-time algorithm [6], where $c_d = 1/((d+1)(\lceil d/2 \rceil + 1))$.

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 $\{a, \ldots, b\}$.

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.

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. □

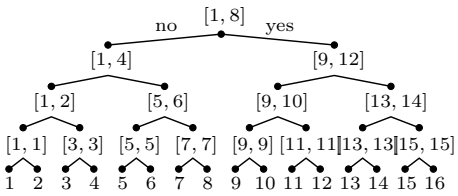

Fig. 4. The binary search tree $\mathcal{B}$. A label $[a, b]$ means that the test $q \in D_a \cap D_{a+1} \cap \ldots \cap D_b$ is performed for a query point q.

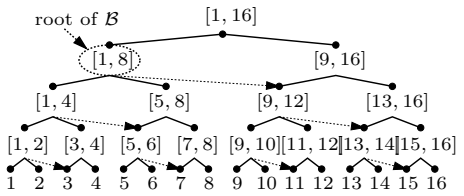

Fig. 5. The auxiliary tree $\mathcal{B}'$ for the construction of $\mathcal{B}$ (see dotted arrows).

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 [7]. 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* MSST *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 MSST *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 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) without taking into account the edge xy. We assume $r_x \geq r_y$.

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 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 $O(n^2 \log n)$ 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) without taking into account the edge xy. Note that $r'_x = |xf_{xy}|$ and $r'_y = |yf_{yx}|$.

Now we compare the diameter of $\mathcal{T}_{xy}$ with that of $\mathcal{T}_{\text{di}}$. We have $\max\{r'_x, r'_y\} \leq r_x$. 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 $\operatorname{diam} \mathcal{T}_{xy} = r'_x + \delta + r'_y \leq 2\max\{r'_x, r'_y\} + \delta \leq 2r_x + \delta$. Now we define two constants α and β that only depend on $\mathcal{T}_{\text{di}}$. Let $\alpha = \delta/(r_x + r_y)$ and $\beta = r_x/r_y$. Note that $\alpha > 0$ and $\beta \geq 1$. Introducing α and β yields

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

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.

Now we apply our α-β-analysis to $\mathcal{T}_{\text{mono}}$. The stability lemma $r_x < \delta + r_y$ [10] implies that all points in P are contained in the disk $D_{x,\delta+r_y}$ of radius $\delta + r_y$ centered at x. 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 $\operatorname{diam} \mathcal{T}_{\text{mono}} \leq \operatorname{diam} \mathcal{T}$ since $\mathcal{T}_{\text{mono}}$ is the MDmST of P. Thus

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

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

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

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 $\beta \leq g_{\text{equal}}(\alpha) := \frac{\alpha+2}{2-\alpha}$. 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 $f_{\text{bisect}}(\alpha, 1) \equiv 1$.
2. Note that the stability lemma $r_x \leq \delta + r_y$ is equivalent to $\beta \leq g_{\text{stab}}(\alpha) := \frac{\alpha+1}{1-\alpha}$. Along g_{stab} the tree $\mathcal{T}_{\text{mono}}$ is optimal since there $f_{\text{mono}} \equiv 1$.
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.

The partial derivatives show 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 scheme uses a grid, the second and third 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 [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 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 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 $O(E) \times O(E)$ cells over P, choose an arbitrary representative point for each cell and use the exact algorithm of Ho et al. [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 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 4. 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 [10].

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 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 [6] 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 [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 5. *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 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) 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.

Observation 1 *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 5(ii) $|uv| \leq (1 + 4/\tau)|pq|$ and by Lemma 5(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

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

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 [10].

Lemma 6. *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.*

4.4 A fast approximation scheme

Now we describe a more involved algorithm. It is faster than the previous algorithm for $n = \Omega(E)$. The correctness proof is in the full version [9, 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 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.

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

Algorithm 1 Approx-MDdST(P, ε)

Ensure: diam $\mathcal{T} \leq (1+\varepsilon)\, d_{\min}$
Phase 1: initializing
1: choose $\kappa \in (0, \min\{0.9\varepsilon, 1/2\})$; set $\gamma \leftarrow \lceil 4/\kappa \rceil$
2: **for** $i \leftarrow 1$ **to** γ **do**
3: $l_i \leftarrow$ line with angle $i\frac{\pi}{\gamma}$ to the horizontal
4: $F_i \leftarrow l_i$-ordering of P
5: **end for** i
6: **for** $i \leftarrow$ **to** γ **do**
7: rotate P and l_i such that l_i is horizontal
8: let $p_1, \ldots, p_n$ be the points in F_i from left to right
9: $d_i \leftarrow |p_1.x - p_n.x|$
10: **for** $j \leftarrow 1$ **to** γ **do**
11: $b_{ij} \leftarrow$ point on l_i at dist. $j\frac{d_i}{\gamma+1}$ to the right of p_1
12: **for** $k \leftarrow 1$ **to** γ **do**
13: $L'_{ijk} \leftarrow l_k$-ordered subset of F_k to the left of b_{ij}
14: $R'_{ijk} \leftarrow l_k$-ordered subset of F_k to the right of b_{ij}
15: **end for** k
16: **end for** j
17: **end for** i
Phase 2: computing approximate farthest neighbors
18: **for** $i \leftarrow 1$ **to** γ **do**
19: **for** $j \leftarrow 1$ **to** γ **do**
20: **for** $k \leftarrow 1$ **to** n **do**
21: $N(p_k, i, j, L) \leftarrow p_k$ {dummy}
22: **for** $l \leftarrow 1$ **to** γ **do**
23: $p_{\min} \leftarrow$ first point in L'_{ijl}
24: $p_{\max} \leftarrow$ last point in L'_{ijl}
25: $N(p_k, i, j, L) \leftarrow$ the point in $\{p_{\min}, p_{\max}, N(p_k, i, j, L)\}$ furthest from p_k
26: **end for** l
27: **end for** k
28: repeat lines 20–27 with R instead of L
29: **end for** j
30: **end for** i
Phase 3: testing pole candidates
31: $\tau = 8(\frac{1+\varepsilon}{(1+\varepsilon-(1+\kappa)(1+\kappa/24)} - 1)$
32: build WSPD for P with separation constant τ
33: $d \leftarrow \infty$ {smallest diameter so far}
34: **for** each pair (A, B) in WSPD **do**
35: choose any two points $u \in A$ and $v \in B$
36: find l_i with the smallest angle to the line through u and v
37: $D \leftarrow \infty$ {approximate diameter of tree with poles u and v, ignoring $|uv|$}
38: **for** $j \leftarrow 1$ **to** γ **do**
39: $D \leftarrow \min\{D, |N(u, i, j, L)u| + |vN(v, i, j, R)|, |N(u, i, j, R)u| + |vN(v, i, j, L)|\}$
40: **end for** j
41: **if** $D + |uv| < d$ **then** $u' \leftarrow u$; $v' \leftarrow v$; $d \leftarrow D + |uv|$ **end if**
42: **end for** (A, B)
43: compute $\mathcal{T} \leftarrow \mathcal{S}_{u'v'}$
44: **return** $\mathcal{T}$

$O(\gamma^3 n + \gamma n \log n)$, see lines 1–17 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 we describe 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. $\tau = \Theta(1/\varepsilon)$. 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 $\mathcal{S}_{uv} - |uv|$, see [9, Section 4.5].

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 41–51 in Algorithm 1) 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 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 1 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. [10] in $O^*(E^2 + n)$ time using $O(E^2 + n)$ space. The tree with smaller diameter is a $(1+\varepsilon)$-approximation of the MDST of P. □

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 [6] and Timothy Chan for sending us an updated version of [6].

References

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

[2] P. K. Agarwal, M. Sharir, and E. Welzl. The discrete 2-center problem. *Discrete & Computational Geometry*, 20, 1998.

[3] G. Barequet and S. Har-Peled. Efficiently approximating the minimum-volume bounding box of a point set in three dimensions. In *Proc. 10th Annual ACM-SIAM Symp. on Discr. Algorithms (SODA'99)*, pages 82–91, Baltimore, 1999.

[4] P. B. Callahan and S. R. 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, Jan. 1995.

[5] T. 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] T. M. Chan. Semi-online maintenance of geometric optima and measures. In *Proc. 13th Symp. on Discr. Algorithms (SODA'02)*, pages 474–483, 2002.

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

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

[9] J. Gudmundsson, H. Haverkort, S.-M. Park, C.-S. Shin, and A. Wolff. Approximating the geometric minimum-diameter spanning tree. Technical Report 4/2002, Institut für Mathematik und Informatik, Universität Greifswald, Mar. 2002. See `www.uni-greifswald.de/~wwwmathe/preprints/shadow/wolff02_4.rdf.html`.

[10] J.-M. Ho, D. T. Lee, C.-H. Chang, and C. K. Wong. Minimum diameter spanning trees and related problems. *SIAM Journal on Computing*, 20(5):987–997, 1991.

[11] M. J. Spriggs, J. M. Keil, S. Bespamyatnikh, M. Segal, and J. Snoeyink. Computing a $(1+\epsilon)$-approximate geometric minimum-diameter spanning tree. Private communication, 2002.