A Fast and Simple Algorithm for Computing Approximate Euclidean Minimum Spanning Trees

Sunil Arya*
Dept. of Computer Science and Engineering
The Hong Kong University
of Science and Technology

David M. Mount†
Dept. of Computer Science and
Inst. for Advanced Computer Studies
University of Maryland

Abstract

The Euclidean minimum spanning tree (EMST) is a fundamental and widely studied structure. In the approximate version we are given an n-element point set P in $\mathbb{R}^d$ and an error parameter $\varepsilon > 0$, and the objective is to compute a spanning tree over P whose weight is at most $(1+\varepsilon)$ times that of the true minimum spanning tree. Assuming that d is a fixed constant, existing algorithms have running times that (up to logarithmic factors) grow as $O\big(n/\varepsilon^{\Omega(d)}\big)$. We present an algorithm whose running time is $O\big(n\log n + \big(\varepsilon^{-2}\log^2\frac{1}{\varepsilon}\big)n\big)$. Thus, this is the first algorithm for approximate EMSTs that eliminates the exponential ε dependence on dimension. (Note that the O-notation conceals a constant factor of the form $O(1)^d$.) The algorithm is deterministic and very simple.

Keywords: Euclidean minimum spanning trees, well-separated pair decompositions, approximation algorithms.

1 Introduction

Given a set P of n points in $\mathbb{R}^d$, the Euclidean minimum spanning tree (EMST) of P is the minimum spanning tree of the complete graph on P where the weight of each edge is the Euclidean distance between its two points. This is a fundamental mathematical structure, which has numerous applications. The problem of computing EMSTs has been extensively studied. Throughout, we assume that the dimension d is a fixed constant.

A straightforward solution involves constructing the complete Euclidean graph on P and computing the MST of this graph in $O(n^2)$ time. In one of the earliest papers in the field of computational geometry, Shamos and Hoey proved that the EMST is a subgraph of the Delaunay triangulation, which leads to an $O(n\log n)$ time algorithm for $d = 2$ [12]. Yao showed that the problem could be solved in subquadratic time in $\mathbb{R}^d$ for any constant d [14], and the best known algorithm by Agarwal *et al.* runs in time roughly $O\big(n^{2-2/(\lceil d/2\rceil+1)}\big)$ [1], which is little better than quadratic except in very small dimensions.

This has led to consideration of approximation algorithms. Given an approximation parameter $\varepsilon > 0$, an ε-approximate EMST is any spanning tree on P whose total weight is larger than the exact EMST by a factor of at most $1+\varepsilon$. Vaidya's results on spanner graphs imply an $O(\varepsilon^{-d} n\log n)$ time solution [13]. We refer to quantity d in the ε^{-d} term as the algorithm's *exponential ε dependence*. In spaces of moderate dimension (say $5 \le d \le 20$) this dependence is a significant practical component of the algorithm's running time. Callahan and Kosaraju showed that the exponential dependence on dimension could be reduced roughly by one half. In [6] they introduced the important concept of a well-separated pair decomposition (WSPD), and in [5] they showed how to apply WSPDs together with a more efficient approach to approximating bichromatic closest pairs to obtain an algorithm with running time $O\big(n\log n + \big(\varepsilon^{-d/2}\log\frac{1}{\varepsilon}\big)n\big)$. Recently, through an improvement to the discrete Voronoi diagram data structure, Arya and Chan further reduced the dimensional dependence, obtaining a randomized algorithm that runs in expected time $O\big(\varepsilon^{-(d/3+O(1))} n\log n\big)$ [3]. The problem has also been considered in high-dimensional spaces, where d is treated as an asymptotic quantity. Har-Peled *et al.* [11] have shown that ε-approximate EMSTs can be computed in time $O(dn^{2-O(\varepsilon)}\varepsilon^{-1}\log^3 n)$. (See also Borodin *et al.* [4].)

Another direction involves algorithms for approximating just the weight of the spanning tree. This originated with an algorithm due to Chazelle *et al.* for computing an ε-approximation to the weight of the MST of a graph [7]. Czumaj *et al.* adapted this to Euclidean space, showing how to compute an ε-approximation to

*Research supported by the Research Grants Council of Hong Kong, China under project number 16200014. Email: arya@cse.ust.hk.

†Research supported by NSF grant CCF-1117259 and ONR grant N00014-08-1-1015. Email: mount@cs.umd.edu.

the weight of the EMST in time $\widetilde{O}\big(\sqrt{n}\varepsilon^{-(d/2+O(1))}\big)$ (with high probability) [8]. Their algorithm assumes free access to an oracle for answering orthogonal range emptiness queries and approximate nearest neighbor queries. Czumaj and Sohler presented an algorithm that operates in any metric space (assuming access to a distance oracle) that computes a weight approximation to the MST in time $\widetilde{O}(n/\varepsilon^7)$ [9].

In summary, after almost 25 years of study, all known near-linear time algorithms for the ε-approximate spanning tree problem have exponential ε dependencies that grow linearly with dimension. In this paper we show that these dimensional exponential ε dependencies can be eliminated altogether. Our main result is an algorithm, which given a set of n points in $\mathbb{R}^d$, computes an ε-approximate EMST in time $O\big(n\log n + \big(\varepsilon^{-2}\log^2\frac{1}{\varepsilon}\big)n\big)$ and space $O(n)$. (Recall that we assume that d is a constant, and the O-notation conceals exponential factors of the form $O(1)^d$.) Our algorithm is both simple and deterministic. In particular, it relies on the same quadtree-based technology used in Vaidya's 1991 algorithm.

Our algorithm is based on a simple, almost trivial, modification of a WSPD-based algorithm presented by Callahan and Kosaraju [5]. First, the algorithm computes a 2-WSPD for the point set (see Section 2.3 for definitions). The WSPD consists of $O(n)$ pairs. Their algorithm extracts an ε-approximate bichromatic pair from each well-separated pair in roughly $O\big(\varepsilon^{-d/2}\big)$ time and then returns the MST of the resulting graph. It is shown in [5] that the result is an ε-approximate EMST. Our modification exploits the following insight. If it takes longer than $O(\varepsilon^{-2})$ time to compute the approximate closest pair, we can infer that there is considerable weight in the EMST in the vicinity of this closest pair. If so, we allow for a larger approximation error when computing this edge, and we charge the error to EMST edges in the vicinity of the well-separated pair. While the modification is very simple, our best analysis of the algorithm's running time is more involved. We employ a quadtree-based charging argument that shows that the charges assessed to each edge of the spanning tree can be bounded by a geometric series that decays as a function of the level of the node that assesses the charge.

In Section 2, we present some preliminary definitions and observations. In Section 3, we present our algorithm and discuss its running time as a function of a parameter γ. In Section 4, we present a simple (but suboptimal) analysis of the algorithm's performance, which results by setting $\gamma = O(\log\frac{n}{\varepsilon})$. In Section 5, we present a more sophisticated analysis of the algorithm's performance, which results by setting $\gamma = O(\log\frac{1}{\varepsilon})$.

2 Preliminaries

In this section we present a number of definitions and preliminary observations, which will be useful later. Recall that the input to our algorithm is an n-element point set P in $\mathbb{R}^d$ and an approximation factor $\varepsilon > 0$.

2.1 Preconditioning

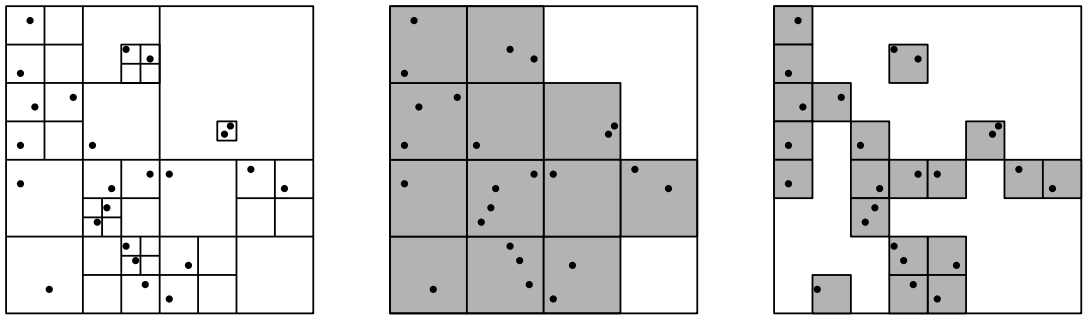
It will simplify the presentation of the algorithm to begin by preconditioning the input set. First, we apply a uniform scaling and translation so that P lies within the unit hypercube $[0,1]^d$ and $\mathrm{diam}(P) \geq 1$. (This is done by computing the smallest axis-parallel hypercube containing P and then scaling and translating it to lie within the unit hypercube. After computing the MST, we can simply apply the inverse of this transformation.)

Our algorithm is based on a compressed quadtree decomposition of P (described below). The basic analysis of Section 4 is sensitive to the height of the tree. A common way to control the height of a quadtree tree is to perturb the points into a nice configuration in a manner that does not significantly alter the weight of the MST (see, e.g., [2]). For an appropriate constant c, depending on the dimension, we round each point of P to the closest point of a quadtree-aligned grid of side length $s_{\min} = c\varepsilon/n$. The value of c may be chosen[1] so that the weight of the MST of the perturbed point set is larger by a factor of at most $1 + \varepsilon/2$. We can easily compensate for this additional error by invoking our algorithm with the approximation parameter decreased by a constant factor. Since this will not affect our asymptotic time complexity, we may assume henceforth that the point set has been perturbed in this manner, and ε has been appropriately modified. Clearly, this preconditioning can be performed in $O(n)$ time. Note that this rounding process is merely a conceptual convenience, since our full analysis of Section 5 does not make any assumptions on the height of the quadtree.

2.2 Compressed Quadtrees and Grid Hierarchies

Our next step is to store the n-element point set P in a compressed quadtree. For the sake of completeness, let us recall the basic definitions here. Starting with the unit hypercube, a *quadtree box* is defined recursively as any hypercube that can be formed by bisect-

[1] To see this, observe that as a result of rounding, the distance between each pair of points undergoes an additive change of $O(c\varepsilon/n)$. Thus, rounding increases the weight of the MST by an additive term of $O(c\varepsilon)$. Since $wt(\mathrm{MST}(P)) \geq \mathrm{diam}(P) \geq 1$, it is possible to choose c so that the perturbed weight is at most $(1+\varepsilon/2)\cdot wt(\mathrm{MST}(P))$.



(a) (b) (c)

Figure 1: (a) The space partition defined by the compressed quadtree $Q(P)$ of P, (b) a set U of generalized nodes of level 2 that cover P, and (c) the result of expand(U).

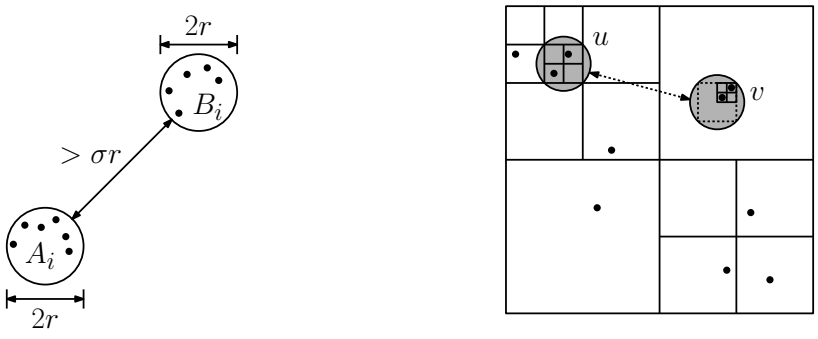
ing an existing quadtree box into 2^d hypercubes, each of half the side length. Repeating such splitting operations results in a rooted 2^d-ary partition tree, called a *quadtree*, where each node is associated with a quadtree box, called its *cell*. We can store P in a quadtree by applying splitting operations until each cell contains at most one point of P. We also assume that each node u whose cell contains at least one point of P is associated with an arbitrary one of these points, called its *representative*.

Unfortunately, (even after preconditioning) the size of the quadtree may not be $O(n)$. This is because many repeated splitting operations would be needed to separate two points that are very close to each other. In such cases we may have long *trivial paths*, where all the points of P lie within the same child cell. To remedy this we compress each maximal trivial path into a single edge, and store a single pointer to the first descendant node where a nontrivial split of P occurs. This is called a *compressed quadtree*. It is possible to build such a structure with $O(n)$ nodes in time $O(n \log n)$ [10]. We denote this tree by $Q(P)$ (Fig. 1(a)). Because the cell associated with each child is at most half the side length of its parent, if preconditioning is applied then $Q(P)$ is of height $O(\log \frac{n}{\varepsilon})$. If not, the height can be as large as $O(n)$.

For $k \geq 0$, the quadtree boxes of side length $1/2^k$ partition the unit hypercube into a grid. The boxes of this grid that contain at least one point of P are said to be *nonempty* (see Fig. 1(b)). We would like to think of the compressed quadtree as providing efficient access to all the nonempty boxes of an infinite hierarchy of such grids. In particular, we would like to associate each nonempty quadtree box b with a corresponding node of $Q(P)$ whose cell is b. Unfortunately, this is not generally possible. This is either because a point lies within a leaf cell of strictly larger side length or because of a compression from a strictly larger parent cell to a strictly smaller child cell. Nonetheless, with a minor enhancement it is possible to achieve this impression. We define a *generalized node* to be a pair consisting of a node u of $Q(P)$ and an integer k, where either u is a leaf whose cell is of side length at least $1/2^k$ or u is an internal node whose cell's side length is at least $1/2^k$ and whose children have side lengths that are strictly smaller than $1/2^k$. The *cell* associated with generalized node (u, k) is a quadtree box of size $1/2^k$ that contains all the points of P that lie within u's cell. A generalized node (u, k) is said to reside at *level* k. It is easy to see that for any $k > 0$, there exists a set of generalized nodes of level k that cover P (see Fig. 1(b) and (c)).

Given a set U of generalized nodes at level k, define the operation expand(U) to return a minimal set of generalized nodes at level $k + 1$ that covers the same points of P as does U. Given $Q(P)$, this operation can easily be performed in time $O(|U|)$. In particular, for each $(u, k) \in U$, if u is a leaf, we replace (u, k) with $(u, k + 1)$ (thus contracting the cell about this point). If u is a standard internal node, we replace (u, k) with the set $(u', k + 1)$, for each nonempty child u' of u. If u is a compressed internal node there are two cases. If u's child cell is of size smaller than $1/2^{k+1}$, we replace (u, k) with $(u, k + 1)$ (again, contracting the cell about the child's cell). Otherwise, we replace it with $(u', k+1)$, where u' is u's child.

Let $P(u)$ denote the points of P contained within u's cell. Given a generalized node (u, k), we define its *neighborhood* to be the up to 3^d nonempty generalized nodes (including u if it is nonempty) at level k whose boundaries intersect u's cell. Each such node is said to be a *neighbor* of (u, k). To avoid overburdening the terminology, henceforth all references to the "quadtree"



(a) (b)

Figure 2: (a) A σ-well-separated pair and (b) the quadtree-based representation.

will refer to the compressed quadtree $Q(P)$. When k is clear from context, we will refer to the generalized node (u, k) simply as u. We let $\ell(u) = k$ denote its level, and $s(u) = 1/2^k$ denote its side length.

2.3 Well-Separated Pair Decomposition

Our algorithm will make use of a well-separated pair decomposition (WSPD) for P. Given a parameter $\sigma \geq 1$, called the *separation factor*, two sets A and B are *σ-well separated* if they can each be enclosed within balls of some radius r such that the closest distance between these balls is greater[2] than σr (see Fig. 2(a)). Given an n-element point set P in $\mathbb{R}^d$ and $\sigma > 0$, define a *σ-well-separated pair decomposition* of P (*σ-WSPD*) to be a collection of pairs $\Psi(P) = \{\{A_1, B_1\}, \ldots, \{A_k, B_k\}\}$ of nonempty subsets of P such that

(1) for any $\{A, B\} \in \Psi(P)$, A and B are σ-well separated, and

(2) for any two distinct points $p, q \in P$, there exists exactly one pair $\{A, B\} \in \Psi(P)$ such that p lies in one of these sets, and q lies in the other.

Callahan and Kosaraju introduced WSPDs, presented an efficient construction algorithm, and discussed a number of applications [6]. The following lemma summarizes the properties of the WSPD that will be relevant here. The proof follows by a straightforward modification of the construction and analysis given by Har-Peled [10].

Lemma 2.1. *Given an n-element point set P in $\mathbb{R}^d$ and any $\sigma \geq 1$, it is possible to build a σ-WSPD $\Psi(P)$ of size $O(\sigma^d n)$ in time $O(n \log n + n\sigma^d)$, such that:*

(i) *Each well-separated pair of $\Psi(P)$ is represented by a pair of (generalized) quadtree nodes (u, v) of the same level. The associated pair is $(P(u), P(v))$, and the separating balls are the minimum balls enclosing u and v's cells (see Fig. 2(b)).*

(ii) *Each (generalized) node u of the quadtree occurs in $O(\sigma^d)$ distinct pairs of $\Psi(P)$.*

For our purposes, it suffices to use a WSPD with separation factor 2. Such a WSPD will have size $O(n)$, it can be computed in $O(n \log n)$ time, and each node occurs in $O(1)$ pairs. In light of this lemma, we will abuse notation by identifying each well-separated pair as a pair (u, v) of quadtree nodes. The cells of u and v are called the *dumbbell heads* of the well-separated pair. WSPDs have a number of useful properties with respect to MSTs and approximate MSTs, as shown in the following lemma due to Callahan and Kosaraju.

Lemma 2.2. (Callahan and Kosaraju [5]) *Given a point set P and a σ-well-separated pair decomposition $\Psi(P)$ for any $\sigma \geq 2$:*

(i) *For each pair $(u, v) \in \Psi(P)$, there is at most one edge of $P(u) \times P(v)$ in $\mathrm{MST}(P)$.*

(ii) *For each pair $(u, v) \in \Psi(P)$ that contributes an edge to $\mathrm{MST}(P)$, let (p, q) be any pair of points from $P(u) \times P(v)$. Then these edges form a spanning tree of P.*

(iii) *If p and q from (ii) are chosen so that $\|pq\| \leq (1 + \varepsilon) \cdot \mathrm{dist}(P(u), P(v))$, then this spanning tree is an ε-approximate MST of P.*

3 The Algorithm

Recall that we are given a set P of n points in $\mathbb{R}^d$ and an approximation parameter $\varepsilon > 0$. Our objective is to

[2] In contrast to standard definitions, we require that the separation distance be strictly larger than σ times the enclosing radii.

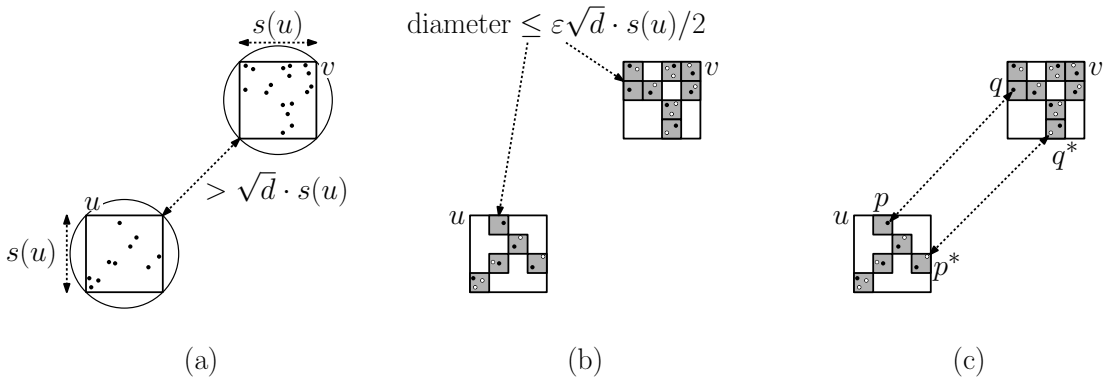


Figure 3: Computing an ε-approximate closest pair.

compute a spanning tree T of P such that

$$wt(T) \;\leq\; (1+\varepsilon)\cdot wt(\mathrm{MST}(P)),$$

where $\mathrm{MST}(P)$ denotes the Euclidean minimum spanning tree of P, and $wt(T)$ denotes T's total edge weight. Our algorithm follows the general approach of earlier algorithms [5, 13]. Given P, we first construct a 2-well-separated pair decomposition $\Psi(P)$ and then construct a sparse graph G by judiciously selecting a pair of points from each well-separated pair. Given the low separation factor, some care is needed in how the point pair is chosen from each well-separated pair. Our algorithm differs in only this one detail. In prior algorithms the time needed to compute the point pair is $O\big(1/\varepsilon^{\Omega(d)}\big)$. We show that by relaxing the criteria for selecting edges, it is possible to find a suitable pair of points without incurring the exponential dependence on d.

3.1 A Simple (but Slower) Solution

To motivate our algorithm it will be illustrative to consider a simple approach, which can be seen as a recasting of Vaidya's original algorithm in the context of WSPDs. Let $\Psi(P)$ be the aforementioned well-separated pair decomposition. For each well-separated pair $(u, v) \in \Psi(P)$ (see Fig. 3(a)), repeatedly subdivide the dumbbell heads associated with u and v, keeping only the nonempty nodes, until the resulting cells have side length at most $\varepsilon \cdot s(u)/2$ (see Fig. 3(b)). Let U and V denote the resulting sets of cells, which we call *mini-cells*. From among all the representatives of U and V, let p and q be the closest pair. Add the edge (p, q) to G. After doing this for all the well-separated pairs, compute and return $\mathrm{MST}(G)$.

To see why this is correct, consider the well-separated pair (u, v), and let (p^*, q^*) be the actual closest pair of points from $P(u) \times P(v)$ (see Fig. 3(c)). By our choice of the separation factor, $\|p^*q^*\| > \sqrt{d}\cdot s(u)$. Because each mini-cell's side length is at most $\varepsilon \cdot s(u)/2$, its diameter is at most $\varepsilon\sqrt{d}\cdot s(u)/2$. The absolute error between the closest representative pair $\|pq\|$ and $\|p^*q^*\|$ is at most twice this diameter, one for each dumbbell head. Therefore

$$\begin{aligned} \|pq\| &\leq \|p^*q^*\| + \varepsilon\sqrt{d}\cdot s(u) \\ &< \|p^*q^*\| + \varepsilon\cdot\|p^*q^*\| \\ &= (1+\varepsilon)\cdot \mathrm{dist}(P(u), P(v)). \end{aligned} \tag{3.1}$$

By Lemma 2.2(iii), $\mathrm{MST}(G) \leq (1+\varepsilon)\cdot\mathrm{MST}(P)$.

The number of mini-cells generated for each well-separated pair can be as large as $\Omega(1/\varepsilon^d)$, and hence the time to construct G is $\Omega(n/\varepsilon^d)$.

3.2 Improved Algorithm

As mentioned in the introduction, our improved algorithm involves a minor twist to the simple solution. Observe that the cases of well-separated pairs that take the greatest time to process are those in which there are many mini-cells. Intuitively, if there are many mini-cells within one of the dumbbell heads, then we can infer that the weight of the minimum spanning tree in the neighborhood of this dumbbell head must be large. (We will make this intuition precise in Lemma 4.3 below.) Rather than charging the error to the MST edge going between this pair, we can instead charge the error to the edges of the MST lying in the vicinity of this dumbbell head. By keeping track of the number of mini-cells being generated, we can terminate the expansion process as soon as the local weight of the spanning tree is large enough to pay for the approximation error. We shall show that the number of mini-cells needed to achieve an ε-approximation grows as $O(1/\varepsilon)$, not $O\big(1/\varepsilon^{O(d)}\big)$.

To implement this we introduce a second stopping criterion to the decomposition process. Let $\gamma \geq 1$ denote a parameter whose value will be fixed later in the analysis. (For our best bound, $\gamma = O(\log \frac{1}{\varepsilon})$.)

Algorithm 1: Approx-MST(P, ε)

```
1  Q(P) ← a quadtree for P;
2  Ψ(P) ← a 2-WSPD of P;
3  G ← (P, ∅);                                   ▷ initialize G to an empty graph over P
4  foreach ((u, v) ∈ Ψ(P)) do                    ▷ each WSP contributes an edge to G
5      U ← {u};  V ← {v}                         ▷ start with the pair's dumbbell heads
6      k ← 1;
7      while (k ≤ λ(ε) and max(|U|, |V|) ≤ γ/ε) do   ▷ expand until depth or node-count satisfied
8          U ← expand(U);  V ← expand(V);
9          k ← k + 1;
10     C ← {(rep(u'), rep(v')) : (u', v') ∈ U × V};   ▷ collect all pairs of representatives
11     (p, q) ← argmin_{(p,q)∈C} ‖pq‖;
12     add edge (p, q) to G;                     ▷ add the closest such pair to G
13 return MST(G);
```

We repeatedly expand the nonempty nodes of each dumbbell head until either:

(i) the cells have side length at most $\varepsilon \cdot s(u)/4$, or

(ii) the number of nonempty nodes exceeds γ/ε.

By condition (ii), the number of expanded nodes per well-separated pair no longer involves exponential dependencies on the dimension, but clearly the approximation error on a per-pair basis is potentially much higher than before.

To control the diameters of the cells of the decomposition, define $\lambda(\varepsilon) = \lceil \log_2 \frac{4}{\varepsilon} \rceil$. In the while loop we expand nodes within the dumbbell heads through at most this many levels. If we reach this level of the decomposition, the side lengths of the resulting cells are at most $\varepsilon \cdot s(u)/4$, which will satisfy the first termination condition. See Algorithm 1 for the complete details. By Lemma 2.2(ii), the graph G computed by this algorithm is a spanning tree of P.

To establish the running time of this algorithm, observe first that because the separation factor is a constant, the quadtree and WSPD can be computed in $O(n \log n)$ time. There are $O(n)$ well-separated pairs, each of which contributes one edge to G. The time needed to process each well-separated pair (the body of for-loop) is dominated by the time to extract the closest pair (Lines 10 and 11), which is $O((\gamma/\varepsilon)^2)$. Thus, it takes $O(n(\gamma/\varepsilon)^2)$ time to compute the edges of G. Since G has $O(n)$ edges, its MST can be computed in $O(n \log n)$ time by any standard algorithm. Thus, the overall running time is $O(n(\log n + (\gamma/\varepsilon)^2))$. The value of γ will be specified later.

Practical observations: Although it is conceptually simpler to present the algorithm as first computing the WSPD and then visiting all the pairs of the WSPD, these two simple algorithms can be elegantly merged into a single algorithm. This is done by inserting the foreach-loop (Line 4) at the point where the WSPD algorithm detects that a pair of quadtree nodes (u, v) is well-separated. Also, observe that the closest pair of representatives for the nodes of U and V are computed in $O((\gamma/\varepsilon)^2)$ time by brute force. A more practical approach would be to iteratively prune the pairs of $U \times V$ (based on inter-box distances), keeping only those that could contribute to the final closest pair. While this seems to be a good idea for any practical implementation, it does not improve our analysis of the worst-case running time.

4 Basic Analysis

In this section we show that, subject to an appropriate selection of value for γ, the output of Approx-MST(P, ε) is a valid ε-approximation to MST(P). We present a basic analysis here, and we will present a more refined analysis later in Section 5. The main result of this section is given below.

Theorem 4.1. (Basic Analysis) *For any fixed dimension d, algorithm Approx-MST computes an ε-approximate EMST for a set of n points in $\mathbb{R}^d$ in $O(n\varepsilon^{-2} \log^2 \frac{n}{\varepsilon})$ time and $O(n)$ space.*

The algorithm's running time increases with γ, but the approximation analysis requires that γ be set sufficiently larger. In this section we will show that it is possible to choose $\gamma = O(\log \frac{n}{\varepsilon})$ in order to achieve the above running time. (The exact value will be specified later.)

Consider a well-separated pair (u, v) that is processed by the algorithm. Recall that (by our generalized representation of quadtree nodes) these nodes are at the same level of the quadtree, and so their cells have the same size. Suppose that the processing termi-

nates after k iterations of the for-loop of Algorithm 1 by condition (ii), that is, $\max(|U|, |V|) > \gamma/\varepsilon$. We may assume without loss of generality that $|U|$ exceeds the threshold. We say that the pair (u, v) is *dense*, and more specifically it is *dense at level* $\ell(u) + k$. Otherwise we say that the pair is *sparse*. Because the denseness of a pair is determined by one of its two nodes, we sometimes abuse this notation by saying that a node u itself is dense. The mini-cells associated with U (and with V as well) are all of side length $s(u)/2^k$.

In order to analyze the algorithm's approximation ratio, let T^* denote all the edges of MST(P) and let $\Psi^*(P)$ be the subset of well-separated pairs $\Psi(P)$ that contribute an edge to T^*. By Lemma 2.2(i), each element of $\Psi^*(P)$ contributes exactly one edge to T^*. Let T_s^* and T_d^* denote a partition of T^* according to whether the corresponding pair in $\Psi^*(P)$ is sparse or dense, respectively.

The edges generated by sparse pairs behave in the same manner as the edges computed in the simple algorithm of Section 3.1, and the error is charged to the corresponding MST edge. The error induced by each dense pair will be charged to the weight of the spanning tree in the neighborhood of the dense dumbbell head. With this in mind, let us identify the edges to be charged in terms of this classification. Since we add one edge to G for each pair of $\Psi(P)$, there is an edge of G for each pair of $\Psi^*(P)$. Let T_s and T_d denote the edges of G corresponding to the sparse and dense pairs of $\Psi^*(P)$, respectively. Let T denote the subgraph of G resulting from just these edges. (Note that T is not the same as MST(G) but rather consists of the edges of G corresponding to the well-separated pairs of MST(P).) By Lemma 2.2(ii), T is a spanning tree of P. Since T is a connected subgraph of G, we have $wt(\text{MST}(G)) \le wt(T)$. Thus, in order to establish our approximation bound, it suffices to show that $wt(T) \le (1+\varepsilon) \cdot wt(\text{MST}(P))$. Before doing this, we present two useful bounds. The first lemma bounds the error for the sparse well-separated pairs.

LEMMA 4.1. *Given T_s^* and T_s defined above, $wt(T_s) \le \left(1 + \frac{\varepsilon}{2}\right) \cdot wt(T_s^*)$.*

Proof. Consider a sparse pair (u, v). Let $(p^*, q^*) \in T_s^*$ be the edge of MST(P) from this well-separated pair, and let $(p, q) \in T_s$ denote the corresponding pair of representatives chosen by Algorithm 1. By sparseness, each nonempty node of the expansion has side length at most $\varepsilon \cdot s(u)/4$ and hence diameter $\varepsilon\sqrt{d} \cdot s(u)/4$. Following the same reasoning as in Section 3.1 (recall Eq. (3.1)) but with half the diameter value, we obtain $\|pq\| \le \left(1 + \frac{\varepsilon}{2}\right)\|p^*q^*\|$. Summing over all $(p, q) \in T_s$ and the corresponding pairs $(p^*, q^*) \in T_s^*$ establishes the result. □

As mentioned earlier, the analysis of the dense-pair case relies on a charging argument, where the error committed for each edge of T_d is charged to the weight of MST(P) in the vicinity of one of the dumbbell heads of the associated well-separated pair. This can result in charging the same MST edges multiple time. Our second lemma shows that this can be handled by adjusting the value of the parameter γ.

LEMMA 4.2. *Given T_d^* and T_d defined above, we may choose $\gamma = O(\log \frac{n}{\varepsilon})$ such that $wt(T_d) - wt(T_d^*) \le \frac{\varepsilon}{2} \cdot wt(\text{MST}(P))$.*

This second lemma requires more effort to prove, and we will defer its proof until later. Assuming these two lemmas for now, let us complete the analysis of the approximation ratio. As an immediate consequence, we have the following two bounds:

$$wt(T_s) \le wt(T_s^*) + \frac{\varepsilon}{2} \cdot wt(T_s^*) \tag{4.2}$$

$$wt(T_d) \le wt(T_d^*) + \frac{\varepsilon}{2} \cdot wt(\text{MST}(P)). \tag{4.3}$$

The sum of the left-hand sides equals $wt(T)$. Since $wt(T_s^*) \le wt(T_s^*) + wt(T_d^*) = wt(\text{MST}(P))$, the sum of the right-hand sides is at most $(1+\varepsilon)wt(\text{MST}(P))$. Combining this with the running time analysis of the previous section establishes Theorem 4.1.

The remainder of this section is devoted to proving Lemma 4.2. Consider a dense pair (u, v). Let $(p, q) \in T_d$ denote the associated pair of representatives chosen by Algorithm 1, and let (p^*, q^*) be the corresponding pair from T_d^*. Without loss of generality, we may assume that u is the dense node of the pair. Let $\ell(u)+k$ denote the level at which u is dense. Because (p, q) is the closest pair of representatives, and the side lengths of the cells from which they are chosen is $s(u)/2^k$, the absolute error committed by choosing (p, q) is at most twice the cell's diameter, that is,

$$\|pq\| - \|p^*q^*\| \le 2\sqrt{d} \cdot \frac{s(u)}{2^k}.$$

Let $\text{err}(u) = 2\sqrt{d} \cdot s(u)/2^k$ denote this error term. (Note that the value of k will generally differ for each dense pair, but our analysis will be independent of k's value.) The absolute weight error for all the dense pairs is

$$wt(T_d) - wt(T_d^*) = \sum_{(p,q) \in T_d} (\|pq\| - \|p^*q^*\|).$$

Let D denote the set of dense nodes of the quadtree. By Lemma 2.1(ii) there is a constant c_π (depending on

dimension) such that each node $u \in D$ appears in at most c_π distinct pairs of the WSPD, and therefore u contributes at most c_π times to the error. Thus, we obtain

$$wt(T_d) - wt(T_d^*) \;\leq\; c_\pi \cdot \sum_{u \in D} \text{err}(u).$$

Define Σ to be the right-hand side of the above inequality. To complete the proof, it suffices to show that

$$\Sigma \;\leq\; \frac{\varepsilon}{2} \cdot wt(\text{MST}(P)). \tag{4.4}$$

We will employ the following useful lower bound on the weight of the MST based on counting the number of nonempty cells. This was proved by Czumaj *et al.* [8], but for the sake of completeness we present the result in our particular context. Given a cell b of the quadtree, define the *restriction* of $\text{MST}(P)$ to b to be the portion of the MST (viewed as a set of line segments) that intersect b.

LEMMA 4.3. *Consider a quadtree node u whose cell has been decomposed into m nonempty quadtree boxes each of side length s. There exists a constant c (depending on the dimension) and a neighbor v of u such that the weight of* $\text{MST}(P)$ *restricted to v's cell is at least sm/c.*

Proof. Let b denote u's cell, and let B denote the set of m nonempty quadtree boxes of side length s within b (see Fig. 4(a)). It is possible to assign 2^d colors to these boxes in a generalized checkerboard fashion, so that each pair of boxes of the same color is separated by a distance of at least s (see Fig. 4(b)).

One of these color classes contains at least $m/2^d$ boxes. Because the MST is connected, the portions of the edges of the MST that lie outside these boxes form a Steiner tree that connects them. Consider a twice-around tour of this Steiner tree (that is, double each edge and take an Euler tour). Trim each path that travels outside of b to a subpath of length s (see Fig. 4(c)). The result is a set of $m/2^d$ paths that lie entirely within u's neighborhood, where each path is of length at least s. The total weight of these paths is at least $sm/2^d$. Allowing for the double-counting of MST edges along these paths, the weight of the MST within u's neighborhood is at least $sm/2^{d+1}$. Therefore, at least one of the 3^d boxes in b's neighborhood (possibly b itself) contains MST weight at least $sm/\big(2^{d+1}3^d\big)$. Setting $c = 2^{d+1}3^d$ yields the desired bound. □

Since u is dense at level $\ell(u) + k$, it can be decomposed into at least γ/ε nonempty boxes, each of side length $s(u)/2^k$. By the above lemma, there is a neighboring node v such that the weight of the MST restricted to this node's cell is at least

$$\frac{s(u)}{2^k} \cdot \frac{\gamma}{\varepsilon} \cdot \frac{1}{c} \;=\; \frac{\gamma}{2c\varepsilon\sqrt{d}} \cdot \text{err}(u).$$

Letting $wt(v)$ denote the weight of the MST restricted to v's cell, we have $\text{err}(u) \leq 2c\varepsilon\sqrt{d} \cdot wt(v)/\gamma$. Observe that any node v can be charged at most 3^d times in this manner, in particular by the nodes of its neighborhood. Therefore, we have

$$\Sigma \;=\; c_\pi \sum_{u \in D} \text{err}(u) \;\leq\; \frac{2 \cdot 3^d c c_\pi \varepsilon \sqrt{d}}{\gamma} \sum_{v \in Q(P)} wt(v).$$

The sum of $wt(v)$ for all nodes v at any given level of the tree is at most $wt(\text{MST}(P))$. By preconditioning, the smallest cell is of size $\Omega(\varepsilon n)$, and so the tree has $O\big(\log \frac{n}{\varepsilon}\big)$ levels. Thus, there exists a constant c' such that

$$\Sigma \;\leq\; \frac{c'}{\gamma} \cdot \log \frac{n}{\varepsilon} \cdot \frac{\varepsilon}{2} \cdot wt(\text{MST}(P)).$$

Therefore, by setting $\gamma = c' \cdot \log \frac{n}{\varepsilon} = O(\log \frac{n}{\varepsilon})$ we have $\Sigma \leq \frac{\varepsilon}{2} \cdot wt(\text{MST}(P))$, as desired. Earlier, we showed that the algorithm's running time is $O(n(\log n + (\gamma/\varepsilon)^2))$. Given our choice of γ, the running time is $O(n\varepsilon^{-2} \log^2 \frac{n}{\varepsilon})$. This completes the proof of Lemma 4.2 and establishes Theorem 4.1.

5 Full Analysis

We observed in the previous section that the principal issue in analyzing the error from the dense well-separated pairs is that the same edges of the MST are multiply charged. This arises from two sources. First, each cell is contained within a constant number c_π of well-separated pairs. Second, each MST edge could be charged by well-separated pairs arising at any of the $O(\log \frac{n}{\varepsilon})$ levels in the quadtree. Clearly, the latter is the more significant of the two.

The parameter γ in the Approx-MST is adjusted to compensate for this overcharging. This parameter controls the number of mini-cells that are generated before declaring that a cell is dense. Increasing its value results both in greater accuracy and higher processing time. Since the processing time grows quadratically with γ, we would like to keep its value as small as possible. In the previous section we showed that it is possible to choose $\gamma = O(\log \frac{n}{\varepsilon})$. Here we will give a more precise analysis, which shows that it suffices to set $\gamma = O(\log \frac{1}{\varepsilon})$. Here is our main result.

THEOREM 5.1. (FULL ANALYSIS) *For any fixed dimension d, algorithm Approx-MST computes an ε-approximate EMST for a set of n points in $\mathbb{R}^d$ in $O\big(n \log n + \big(\varepsilon^{-2} \log^2 \frac{1}{\varepsilon}\big)n\big)$ time and $O(n)$ space.*

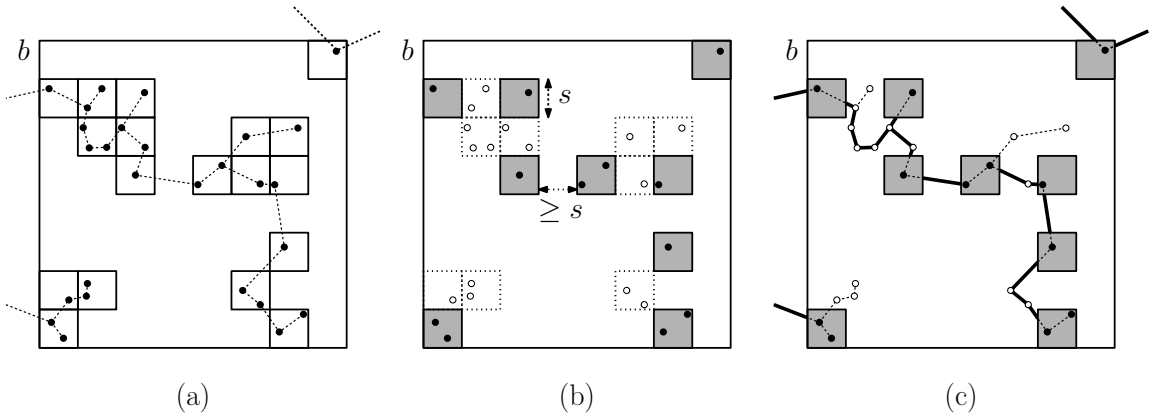


Figure 4: Proof of Lemma 4.3.

The leading $O(n \log n)$ term accounts for computing the quadtree and the WSPD. We will show that once these have been computed, the remaining running time is $O\big((\varepsilon^{-2} \log^2 \frac{1}{\varepsilon})n\big)$. The analysis follows the same structure as in the previous section. In particular, we distinguish between sparse and dense well-separated pairs. The sparse-pair analysis given in Lemma 4.1 is unchanged. The principal improvement is to the dense-pair analysis of Lemma 4.2, which is given in the following lemma.

LEMMA 5.1. *Recalling the definitions of T_d^* and T_d given just prior to Lemma 4.2, we may choose $\gamma = O(\log \frac{1}{\varepsilon})$ such that $wt(T_d) - wt(T_d^*) \leq \frac{\varepsilon}{2} \cdot wt(\text{MST}(P))$.*

The rest of this section is devoted to the proof. Before getting into the details, let us begin with a high-level overview. The analysis begins in the same way as that of Lemma 4.2 up through Lemma 4.3. The remainder involves establishing the upper bound on Σ from Eq. (4.4), but based on the new value of γ.

Recall that at least one of the nodes of each dense well-separated pair (u, v) is dense, and we assume this node to be u. The function $\text{err}(u)$ bounds the absolute error committed at this node. The quantity Σ bounds the sum of errors over all the dense nodes and accounts for the overlap due to multiple well-separated pairs sharing the same dumbbell head. The basic analysis makes the simplifying but pessimistic assumption that the restriction of $\text{MST}(P)$ to any leaf cell may be charged equally by every one of its $O(\log \frac{n}{\varepsilon})$ ancestors. In order for this worst-case to occur, a node must have many ancestors that are dense. We will show that by adjusting the constant factor in the definition of γ, whenever this phenomenon occurs, the charges assessed by successive ancestors decay exponentially with every $O(\log \frac{1}{\varepsilon})$ levels. It will follow that the total charge is dominated asymptotically by just the first $O(\log \frac{1}{\varepsilon})$ ancestors, and this is why the smaller value of γ suffices. Also, because the resulting geometric series converges, it is not necessary to bound the height of $Q(P)$. Hence, the rounding step of the preconditioning is not needed.

In order to apply this level-based analysis of the MST edge weights, we adopt an approach similar to that of Lemma 4.3 by relating the weight of the edges being charged to the sum of the side lengths of the quadtree cells that intersect the edges of $\text{MST}(P)$. We define a new quadtree, called $\overline{Q}(P)$, based on the quadtree boxes that intersect the edges of $\text{MST}(P)$. We then classify a subset of the nodes of $\overline{Q}(P)$ as *bushy*, which correspond roughly to the dense nodes of $Q(P)$. In particular, we show that for each dense node u of $Q(P)$, there is bushy node of $\overline{Q}(P)$ of equal side length among u's neighbors. We demonstrate the exponential drop-off in charges within $\overline{Q}(P)$ and then apply this node correspondence to show that the charging argument applies to $Q(P)$ as well.

Let $s_{\min}$ denote the side length of the smallest leaf cell of $Q(P)$, and consider a quadtree-aligned grid of this side length. Let L denote the cells of this grid that have a nonempty intersection with any edge of $\text{MST}(P)$ (see Fig. 5(a)). Let $\overline{Q}(P)$ denote an (uncompressed) quadtree built over the cells of L (see Fig. 5(b)). Given a node u of $\overline{Q}(P)$, let $L(u)$ denote the elements of L that lie within u's cell. We say that u is *nonempty* if $L(u)$ is nonempty (or equivalently, if u's cell has a nonempty intersection with $\text{MST}(P)$). The principal purpose of introducing $\overline{Q}(P)$ is that we can relate the weight of $\text{MST}(P)$ to the sum of side lengths of the cells of L, which is just $s_{\min} \cdot |L|$. (Note that $\overline{Q}(P)$ is not computed. It is merely used in the analysis.)

LEMMA 5.2. *Given $s_{\min}$ and L defined above, $s_{\min} \cdot |L| \leq 2^{d+1} \cdot wt(\text{MST}(P))$.*

Proof. As in the proof of Lemma 4.3, observe that it

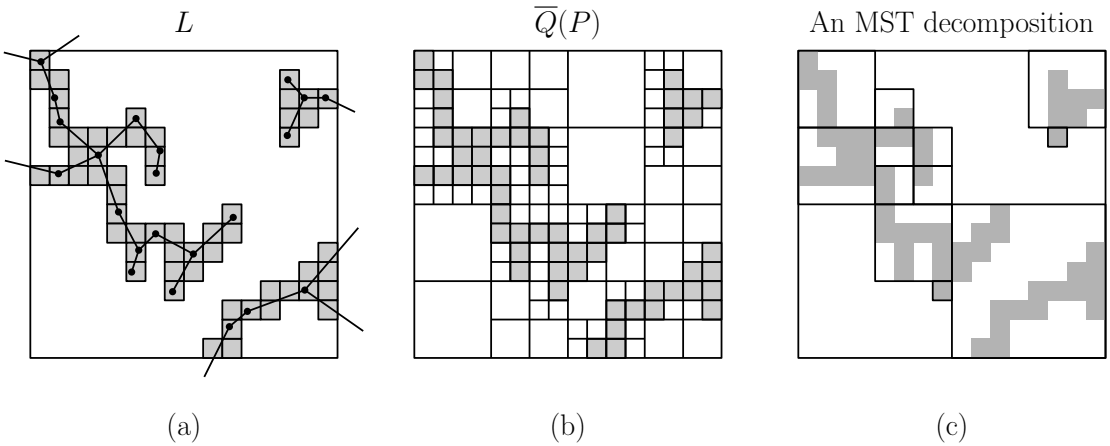


Figure 5: (a) The leaf cells L, (b) the quadtree $\overline{Q}(P)$, and (c) the cells of an MST decomposition.

is possible to assign 2^d colors to the cells of L so that each pair of cells of the same color is separated by a distance of at least $s_{\min}$. At least one of these color classes contains at least $|L|/2^d$ cells. Because the MST is connected, the portions of the edges of the MST that lie outside these cells form a Steiner tree that connects them. The total weight of a twice-around tour of this Steiner tree is at most $2 \cdot wt(\text{MST}(P))$, and each cell is joined to its successor on the tour by a path of length at least $s_{\min}$. Summing up the lengths of these connecting paths, we have $s_{\min} \cdot |L|/2^d \leq 2 \cdot wt(\text{MST}(P))$, from which we conclude that $s_{\min} \cdot |L| \leq 2^{d+1} \cdot wt(\text{MST}(P))$. □

Each nonempty leaf node u of $Q(P)$ intersects MST(P). Because $Q(P)$ and $\overline{Q}(P)$ share the same alignments and by the definition of $s_{\min}$, u's cell contains at least (and possibly equals) one cell of L. (Here, we are thinking of u as a standard leaf node of $Q(P)$, not a generalized node. This is because generalized leaf nodes can be arbitrarily small.) It follows directly that the spatial decomposition defined by $\overline{Q}(P)$ is a refinement of that of $Q(P)$. Because both trees cover the same domain (the unit hypercube), we have the following useful correspondence between the nodes of these trees.

Lemma 5.3. *For every nonempty (generalized) node u of $Q(P)$ whose cell size is at least $s_{\min}$, there exists a nonempty node u' of $\overline{Q}(P)$ such that u and u' share the same cell.*

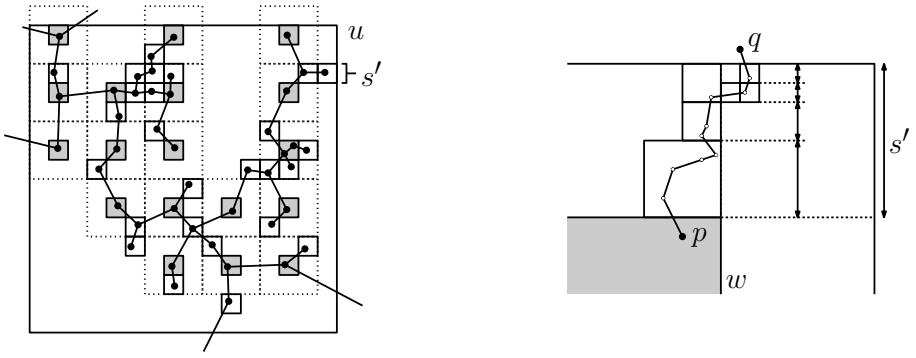
Our analysis of the dense pairs will involve a charging argument. We charge the errors committed by our algorithm to the nodes of $\overline{Q}(P)$, and we then distribute these charges to its leaves L. We will bound the total charge received by any leaf node in terms of its side length $s_{\min}$. For the sake of the charging argument, we focus on nodes that carry a significant amount of weight of the MST, relative to the side length of the node's cell. We will need to apply charging to groups of nodes at various levels with disjoint cells. To do this, we introduce a few useful concepts.

Given a node u of $\overline{Q}(P)$, we define an *MST decomposition* of u to be any subset V of u's nonempty descendants such that the sets $L(v)$ for $v \in V$ form a disjoint cover of $L(u)$. (An example of such a decomposition is illustrated in Fig. 5(c).) Given an MST decomposition V of u, define its *edge weight*, denoted $s(V)$, to be the sum of the side lengths of its corresponding cells, that is, $s(V) = \sum_{v \in V} s(v)$. We say that a node u of $\overline{Q}(P)$ is *chargeable* if for any MST decomposition V of u, $s(V) \geq s(u)/3^d$. Recall from Section 3.2 that $\lambda(\varepsilon) = \lceil \log_2 \frac{4}{\varepsilon} \rceil$. Given a node u and an integer k, where $1 \leq k \leq \lambda(\varepsilon)$, we say that u is *bushy*, and more specifically that it is *bushy at level $\ell(u)+k$*, if the number of chargeable descendants of u at level $\ell(u)+k$ is at least $\gamma/9^d\varepsilon$, and k is the smallest value such that this is true.

The following lemma provides a connection between the dense nodes of $Q(P)$ and the bushy nodes of $\overline{Q}(P)$. It states that for each dense well-separated pair there is a bushy node of $\overline{Q}(P)$ that belongs to the neighborhood of one of this pair's dumbbell heads.

Lemma 5.4. *Let (u, v) be a well-separated pair that is dense, and let u be the dense node of the pair. Let u' be the corresponding node to u in $\overline{Q}(P)$ (which exists by Lemma 5.3). Then there is a bushy node in the neighborhood of u' in $\overline{Q}(P)$. Furthermore, if u is dense at level $\ell(u)+k$ then u' is bushy at level $\ell(u')+k'$, where $k' \leq k$.*

Proof. By definition of denseness, there exists k, $1 \leq k \leq \lambda(\varepsilon)$, such that u has at least γ/ε nonempty



(a) (b)

Figure 6: Proof of Lemma 5.4.

descendants in $Q(P)$ at level $\ell(u) + k$. The cells associated with these nodes are each of side length $s' = s(u)/2^k$ (see Fig. 6(a)).

We can color these nodes using 3^d colors such that any two nodes of the same color have neighborhoods that are disjoint from each other. Clearly, there exists a color class that has at least $\gamma/3^d\varepsilon$ nodes (shaded in Fig. 6(a)). By Lemma 5.3, for each of these nodes there exists a nonempty node in $\overline{Q}(P)$ with the same cell. Let V denote this set of nodes.

We assert that for every $w \in V$, at least one of the nodes within w's neighborhood is chargeable. To see why, observe first that since each w (viewed as a node of $Q(P)$) is nonempty, it contains a point $p \in P$. By the connectivity of the MST, there is a path in the MST that connects p to some point q that lies outside of w's neighborhood. Clearly, the L_∞ distance between p and q must be at least s'. Thus, there exists j, $1 \le j \le d$, such that p and q's jth coordinates differ by at least s' (the vertical dimension in Fig. 6(b)). Consider the subset of cells of any MST decomposition that covers this path. The orthogonal projections of these cells onto the jth coordinate axis cover an interval of length at least s', which implies that the sum of their side lengths is at least s'. Therefore, for any MST decomposition of the 3^d nodes of w's neighborhood, at least one of the nodes has a total edge length of at least $s'/3^d$, and so this node is chargeable. Because the neighborhoods of the nodes of V are disjoint, there are at least $\gamma/3^d\varepsilon$ chargeable nodes.

Note that the chargeable nodes need not be descendants of u, because the neighborhoods of u's descendants may extend into u's neighborhood. Among the 3^d nodes in u's neighborhood, at least one of them contains at least $(\gamma/3^d\varepsilon)/3^d = \gamma/9^d\varepsilon$ chargeable nodes k levels below it. This node satisfies the bushiness condition at level $\ell(u) + k$, and so it is bushy at some level $\ell(u) + k'$, for $k' \le k$. □

Next, we show how to relate the approximation error in $Q(P)$ to the properties of $\overline{Q}(P)$. Consider any node $u \in Q(P)$ that is dense at some level $\ell(u) + k$. By the above lemma, it is associated with a node $u' \in \overline{Q}(P)$ of equal size that is bushy at level $\ell(u') + k'$, where $k' \le k$. Define $\overline{\text{err}}(u') = 2\sqrt{d} \cdot s(u')/2^{k'}$. Note that this is the same as $\text{err}(u')$, but using the bushiness level k' rather than the denseness level k. Let B denote the associated set of bushy nodes of $\overline{Q}(P)$. For each $u' \in B$ there are at most 3^d dense nodes of Q that could have chosen to be associated with u' (those nodes having u' in their neighborhoods). Both err and $\overline{\text{err}}$ are monotonically decreasing functions of level (of denseness and bushiness, respectively), and therefore $\overline{\text{err}}(u') \ge \text{err}(u)$. Recalling the WSPD overlap constant c_π introduced earlier, define $\overline{\Sigma} = c_\pi \sum_{u' \in B} \overline{\text{err}}(u')$. Thus, we have

$$\Sigma \le 3^d \cdot \overline{\Sigma}. \tag{5.5}$$

To complete the analysis of the dense-pair error, it suffices to prove the following bound on $\overline{\Sigma}$.

Lemma 5.5. $\overline{\Sigma} \le \varepsilon \cdot wt(\text{MST}(P))/(2 \cdot 3^d)$.

The remainder of this section is devoted to proving Lemma 5.5. Our approach will be to apply a charge that is proportional to $\overline{\text{err}}(u)$ to each node $u \in B$ and then distribute this charge among a subset of the nonempty leaves of $\overline{Q}(P)$ (that is, the cells of $L(u)$). We will show that for each $w \in L$, the sum of charges assessed to this node is at most proportional to $\varepsilon \cdot s(w)/c$, for a suitable constant c. By summing over all these leaves and appealing to the upper bound of Lemma 5.2, we

will obtain the desired bound provided that c is chosen properly.

For charging purposes, we partition the bushy nodes B into groups. For $0 \leq i < \lambda(\varepsilon)$, define B_i to be the subset of bushy nodes $u \in B$ such that $\ell(u) \equiv i \pmod{\lambda(\varepsilon)}$. Let us fix an arbitrary value of i, and let u_0 be an arbitrary node of B_i. Let $\ell(u_0)+k_0$ be the level at which u_0 is bushy. Define u_0's *charge* to be $s(u_0)/2^{k_0}$, which we denote by $\chi(u_0)$. We propagate this charge to a subset of the nonempty leaves descended from u_0 by the following iterative process. Initially u_0 is the only node with a charge. For any node w that receives a charge from u_0, let $\chi_{u_0}(w)$ denote this charge.

(1) Let u be a bushy descendant of u_0 (possibly u_0 itself) such that $\chi_{u_0}(u) \neq 0$. Let $\ell(u) + k$ be the level at which u is bushy. By definition of bushiness, there are at least $\gamma/9^d\varepsilon$ chargeable descendants of u at level $\ell(u) + k$. Distribute $\chi_{u_0}(u)$ uniformly over these descendants so that each receives a charge of at most $9^d\varepsilon \cdot \chi_{u_0}(u)/\gamma$.

(2) Let v denote any of the descendants of u that receives a charge by rule (1). Distribute this charge to a subset of v's nonempty descendants as follows. Consider the set of paths (including possibly the empty path) that descend from v until first reaching a node of B_i or a nonempty leaf node. Let $V(v)$ denote the subset of nodes at which these paths terminate. (If v itself is in B_i or is a leaf, then $V(v) = \{v\}$.) It is easy to see that $V(v)$ is an MST decomposition of v. Recall that $s(V(v))$ is the sum of side lengths of the boxes of $V(v)$. For each $u \in V(v)$, we assess it a charge of $\chi_{u_0}(v) \cdot s(u)/s(V(v))$.

Observe that each node charged by rule (1) propagates its charge to a subset of its descendants, and each of these nodes then propagates this charge to a subset of its descendants by rule (2). Since each non-leaf node charged by rule (2) is in B_i, this charge will again be propagated by rule (1). It follows that all of u_0's initial charge will be distributed to a subset of its descendant leaves. The following lemma shows that if the value of γ (in the definition of dense nodes) is $\Omega(\log \frac{1}{\varepsilon})$, then the sum of charges distributed to any leaf is proportional to ε times its side length.

LEMMA 5.6. *For any positive constant c', we may choose $\gamma = O(\log \frac{1}{\varepsilon})$ such that for all $w \in L$*

$$\sum_{u \in B} \chi_u(w) \;\leq\; c'\varepsilon \cdot s(w) \;=\; c'\varepsilon \cdot s_{\min}.$$

Proof. Consider any i, $0 \leq i < \lambda(\varepsilon)$, any node $u_0 \in B_i$, and any nonempty leaf node w that receives some of u_0's charge. From the charging rules, there exists a sequence of nodes $\langle u_0, v_1, u_1, v_2, u_2, \ldots, v_m, u_m\rangle$, such that $u_m = w$ and for $1 \leq j \leq m$, v_j receives its charge from u_{j-1} by rule (1), and u_j receives its charge from v_j by rule (2). Note that these nodes are not necessarily distinct, since by rule (2) we may have $u_j = v_j$, but this cannot happen for rule (1), and so $v_{j+1} \neq u_j$.

We will first show that there exists a constant c such that $\chi_{u_0}(w) \leq \varepsilon \cdot s(w)(c/\gamma)^m$. To do this we will show by induction that, for $1 \leq j \leq m$,

$$\chi_{u_0}(v_j) \;\leq\; \varepsilon \cdot s(v_j) \left(\frac{c}{\gamma}\right)^j \frac{1}{3^d} \tag{5.6}$$

$$\chi_{u_0}(u_j) \;\leq\; \varepsilon \cdot s(u_j) \left(\frac{c}{\gamma}\right)^j. \tag{5.7}$$

To start, recall that u_0 is bushy at level $\ell(u_0) + k_0$. By rule (1) u_0 distributes its initial charge of $s(u_0)/2^{k_0}$ uniformly among at least $\gamma/9^d\varepsilon$ descendants, each of which is chargeable and resides k_0 levels below u_0. Since v_1 is one of these nodes, it follows that $s(v_1) = s(u_0)/2^{k_0}$. Therefore, for any $c \geq 27^d$ we have

$$\begin{aligned}\chi_{u_0}(v_1) \;&\leq\; \frac{s(u_0)/2^{k_0}}{\gamma/9^d\varepsilon} \;=\; \frac{\varepsilon \cdot s(v_1) \cdot 9^d}{\gamma} \\ &\leq\; \varepsilon \cdot s(v_1) \left(\frac{c}{\gamma}\right) \frac{1}{3^d}.\end{aligned}$$

Assuming inductively that the bound of Eq. (5.7) holds for v_j, we will show that it also holds for u_j. Let V denote the MST decomposition from charging rule (2). Because v_j is chargeable, we have $s(V) \geq s(v_j)/3^d$. By this rule the charge received by u_j is

$$\begin{aligned}\chi_{u_0}(u_j) \;&=\; \chi_{u_0}(v_j) \cdot \frac{s(u_j)}{s(V)} \\ &\leq\; \left(\varepsilon \cdot s(v_j) \left(\frac{c}{\gamma}\right)^j \frac{1}{3^d}\right) \frac{s(u_j)}{s(v_j)/3^d} \\ &=\; \varepsilon \cdot s(u_j) \left(\frac{c}{\gamma}\right)^j.\end{aligned}$$

Again, assuming inductively that the bound of Eq. (5.6) holds for u_j, for $j < m$, we will show that it also holds for v_{j+1}. Because $j < m$, u_j is not a leaf, and hence $u_j \in B_i$. This implies that u_j is bushy. Let $\ell(u_j) + k_j$ denote the level at which it is bushy. By rule (1), u_j distributes its charge of $\chi_{u_0}(u_j)$ uniformly among at least $\gamma/9^d\varepsilon$ descendants that lie k_j levels below it. By definition, $\lambda(\varepsilon) \leq \lg \frac{1}{\varepsilon} + 3$. By the definition of bushiness (which is derived from the definition of denseness) $k_j \leq \lambda(\varepsilon)$. Therefore,

$$s(v_{j+1}) \;=\; \frac{s(u_j)}{2^{k_j}} \;\geq\; \frac{s(u_j)}{2^{\lambda(\varepsilon)}} \;\geq\; \frac{\varepsilon \cdot s(u_j)}{8}.$$

Therefore, for any $c \geq 8 \cdot 27^d$, we have

$$\begin{aligned}
\chi_{u_0}(v_{j+1}) &\leq \frac{\chi_{u_0}(u_j)}{\gamma/9^d\varepsilon} \\
&\leq \left(\varepsilon \cdot s(u_j)\left(\frac{c}{\gamma}\right)^j\right)\frac{9^d\varepsilon}{\gamma} \\
&\leq \varepsilon \cdot s(v_{j+1})\left(\frac{c}{\gamma}\right)^j \frac{8 \cdot 9^d}{\gamma} \\
&\leq \varepsilon \cdot s(v_{j+1})\left(\frac{c}{\gamma}\right)^{j+1} \frac{1}{3^d}.
\end{aligned}$$

This establishes Eqs. (5.6) and (5.7). Applying this to $w = u_m$, we have $\chi_{u_0}(w) \leq \varepsilon \cdot s(w)(c/\gamma)^m$, as desired.

Next, we will bound the total charge received by w for all the nodes of B_i. Consider two nodes $u_0, u_0' \in B_i$ that distribute a charge to w. Since these are distinct ancestors of w, we may assume that u_0 is a proper ancestor of u_0'. By the above analysis, there exist integers m and m' such that $\chi_{u_0}(w) \leq \varepsilon \cdot s(w)(c/\gamma)^m$ and $\chi_{u_0'}(w) \leq \varepsilon \cdot s(w)(c/\gamma)^{m'}$. We will show that $m > m'$. Let $\langle u_0, v_1, \ldots, u_m\rangle$ and $\langle u_0', v_1', \ldots, u_{m'}'\rangle$ denote the respective sequences of nodes in the above charge-distribution analysis, where $w = u_m = u_{m'}'$. Because both u_0 and u_0' are in B_i, their levels in the quadtree differ by at least $\lambda(\varepsilon)$. In the analysis of the charge distribution for u_0, by rule (1) its first chargeable node v_1 lies at most $\lambda(\varepsilon)$ levels below it, and therefore v_1 is a (not necessarily proper) ancestor of u_0'. The next node in the sequence u_1 is the first descendant of v_1 in B_i, implying that u_1 is a (not necessarily proper) ancestor of u_0'. If $u_1 = u_0'$, then $m' = m - 1$. (This follows from the fact that the nodes selected in the charging process do not depend on any properties of ancestor nodes, just descendants.) If not, we apply the same argument inductively to u_1 until we find the first $j \geq 1$ such that $u_j = u_0'$. It follows that $m' = m - j$, and so $m > m'$ as desired.

Because each node of B_i that charges w has a different power of the (c/γ) term in the charging bound, it follows that the total charge received at w from all these nodes is at most

$$\begin{aligned}
\sum_{u \in B_i} \chi_u(w) &= \sum_{m=1}^{\infty} \varepsilon \cdot s(w)\left(\frac{c}{\gamma}\right)^m \\
&= \varepsilon \cdot s(w) \sum_{m=1}^{\infty}\left(\frac{c}{\gamma}\right)^m.
\end{aligned}$$

If $\gamma \geq 2c$, then the above summation is at most $2c/\gamma$. By summing over all values of i, $0 \leq i < \lambda(\varepsilon)$, we obtain

$$\begin{aligned}
\sum_{u \in B} \chi_u(w) &= \sum_{0 \leq i < \lambda(\varepsilon)} \sum_{u \in B_i} \chi_u(w) \\
&\leq \lambda(\varepsilon) \cdot \varepsilon \cdot s(w) \cdot \frac{2c}{\gamma}.
\end{aligned}$$

By setting $\gamma = 2c\lambda(\varepsilon)/c'$, which is $O(\log \frac{1}{\varepsilon})$ and recalling that $s(w) = s_{\min}$ we achieve the desired bounds. □

By summing up the charges over all the leaves L of $\overline{Q}(P)$ we obtain

$$\sum_{u \in B} \chi(u) \leq c'\varepsilon \cdot s_{\min} \cdot |L|.$$

By combining this with Lemma 5.2, we obtain the following bound on the total charge.

Lemma 5.7. *For any positive constant c', we may choose $\gamma = O(\log \frac{1}{\varepsilon})$ such that*

$$\sum_{u \in B} \chi(u) \leq 2^{d+1}c'\varepsilon \cdot wt(\mathrm{MST}(P)).$$

To complete the proof of Lemma 5.5, recall the WSPD overlap constant c_π (defined just prior to Eq. (4.4)). Select any $c' \leq 1/(c_\pi \cdot 8 \cdot 6^d\sqrt{d})$. For each node $u \in \overline{Q}(P)$ that is bushy at level $\ell(u) + k$, recall that $\overline{\mathrm{err}}(u) = 2\sqrt{d} \cdot s(u)/2^k$ and $\chi(u) = s(u)/2^k$. By the definition of $\overline{\Sigma}$ and applying Lemma 5.7, we have

$$\begin{aligned}
\overline{\Sigma} &= c_\pi \sum_{u \in B} \overline{\mathrm{err}}(u) \\
&= c_\pi \sum_{u \in B} \frac{2\sqrt{d} \cdot s(u)}{2^k} = c_\pi 2\sqrt{d} \sum_{u \in B} \chi(u) \\
&\leq c_\pi 2\sqrt{d} \cdot (2^{d+1}c'\varepsilon) \cdot wt(\mathrm{MST}(P)) \\
&= c'c_\pi \cdot 8 \cdot 6^d\sqrt{d}\left(\varepsilon \cdot \frac{wt(\mathrm{MST}(P))}{2 \cdot 3^d}\right) \\
&\leq \varepsilon \cdot \frac{wt(\mathrm{MST}(P))}{2 \cdot 3^d},
\end{aligned}$$

as desired.

By combining Lemma 5.5 with Eq. (5.5) we obtain

$$\Sigma \leq 3^d \cdot \overline{\Sigma} \leq \frac{\varepsilon}{2} \cdot wt(\mathrm{MST}(P)),$$

which as shown in Eq. (4.4) suffices to complete the error analysis for the dense pairs. By combining this with the analysis of the sparse pairs from Lemma 4.1, we complete the proof of Theorem 5.1.

6 Concluding Remarks

We have presented an algorithm, which given a set of n points in $\mathbb{R}^d$, computes an ε-approximate EMST in time $O\big(n \log n + \big(\varepsilon^{-2} \log^2 \frac{1}{\varepsilon}\big)n\big)$. This is the first algorithm for computing approximate EMSTs (not just the weight) that eliminates all exponential ε dependencies. The algorithm is simple and deterministic, relying on nothing more than a compressed quadtree data structure. Subject to a minor change in some of the algorithm's parameters, it can compute an ε-approximation to the minimum spanning tree in any Minkowski norm (such as the L_1 and L_∞ norms) in the same running time.

Note that our algorithm does not eliminate all exponential dependencies on dimension, because the big-O notation conceals constant factors of the form $O(1)^d$. The analysis of Section 5 suggests that these constants are quite large. We hasten to add that these are worst-case values, and we have made no attempt to optimize their values.

Our algorithm achieves its efficiency by being sloppier in approximating bichromatic closest pairs between well-separated pairs in situations where we can infer there is significant EMST weight in the vicinity of the well-separated pair. This sloppiness implies that certain local properties of the EMST are not necessarily well approximated. For example, it is well known that the exact EMST is the minimum bottleneck spanning tree, meaning that it minimizes the maximum edge length. Our algorithm does not produce an ε-approximation to the bottleneck EMST. (It is easy to show that it produces an $O(\varepsilon^{1/d})$-approximation.) It is an interesting open problem as to whether it is possible to eliminate exponential ε dependencies in computing the approximate the minimum bottleneck spanning tree.

The exact EMST is also known to contain an edge between the closest pair of points of P. It can be shown[3] that our approximate EMST shares this the property. An interesting question is which order statistics of EMST edge lengths can be well approximated without incurring exponential ε dependencies.

7 Acknowledgments

We would like to thank Ahmed Abdelrazek for his comments on an earlier draft of this paper.

[3] This follows from the well-known fact that if p and q are the closest points of P, then in any 2-WSPD of P, there must be a well-separated pair $(\{p\}, \{q\})$ containing just these points. Hence the edge (p, q) will be added to the graph G of Approx-MST.

References

[1] P. K. Agarwal, H. Edelsbrunner, O. Schwartzkopf, and E. Welzl. Euclidean minimum spanning trees and bichromatic closest pairs. *Discrete Comput. Geom.*, 6:407–422, 1991.

[2] S. Arora. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *J. Assoc. Comput. Mach.*, 45:753–782, 1998.

[3] S. Arya and T. M. Chan. Better ε-dependencies for offline approximate nearest neighbor search, Euclidean minimum spanning trees, and ε-kernels. In *Proc. 30th Annu. Sympos. Comput. Geom.*, pages 416–425, 2014.

[4] A. Borodin, R. Ostrovsky, and Y. Rabani. Subquadratic approximation algorithms for clustering problems in high dimensional spaces. In *Proc. 31st Annu. ACM Sympos. Theory Comput.*, pages 435–444, 1999.

[5] P. B. Callahan and S. R. Kosaraju. Faster algorithms for some geometric graph problems in higher dimensions. In *Proc. Fourth Annu. ACM-SIAM Sympos. Discrete Algorithms*, pages 291–300, 1993.

[6] P. B. Callahan and S. R. Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *J. Assoc. Comput. Mach.*, 42:67–90, 1995.

[7] B. Chazelle, R. Rubinfeld, and L. Trevisan. Approximating the minimum spanning tree weight in sublinear time. *SIAM J. Comput.*, 34:1370–1379, 2005.

[8] A. Czumaj, F. Ergün, L. Fortnow, A. Magen, I. Newman, R. Rubinfeld, and C. Sohler. Approximating the weight of the Euclidean minimum spanning tree in sublinear time. *SIAM J. Comput.*, 35:91–109, 2005.

[9] A. Czumaj and C. Sohler. Estimating the weight of metric minimum spanning trees in sublinear time. *SIAM J. Comput.*, 39:904–922, 2009.

[10] S. Har-Peled. *Geometric Approximation Algorithms*. American Mathematical Society, Providence, Rhode Island, 2011.

[11] S. Har-Peled, P. Indyk, and R. Motwani. Approximate nearest neighbor: Towards removing the curse of dimensionality. *Theory of Computing*, 8:321–350, 2012. Article 14.

[12] M. I. Shamos and D. Hoey. Closest-point problems. In *Proc. 16th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 151–162, 1975.

[13] P. M. Vaidya. A sparse graph almost as good as the complete graph on points in K dimensions. *Discrete Comput. Geom.*, 6:369–381, 1991.

[14] A. C. Yao. On construction minimum spanning trees in k-dimensional space and related problems. *SIAM J. Comput.*, 11:721–736, 1982.