Approximate Geometric MST Range Queries

Sunil Arya[*1], **David M. Mount**[†2], **and Eunhui Park**[2]

1 **Department of Computer Science and Engineering**
The Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong, China
arya@cse.ust.hk

2 **Department of Computer Science**
University of Maryland
College Park, Maryland 20742, USA
{mount,ehpark}@cs.umd.edu

Abstract

Range searching is a widely-used method in computational geometry for efficiently accessing local regions of a large data set. Typically, range searching involves either counting or reporting the points lying within a given query region, but it is often desirable to compute statistics that better describe the structure of the point set lying within the region, not just the count.

In this paper we consider the geometric minimum spanning tree (MST) problem in the context of range searching where approximation is allowed. We are given a set P of n points in $\mathbb{R}^d$. The objective is to preprocess P so that given an admissible query region Q, it is possible to efficiently approximate the weight of the minimum spanning tree of $P \cap Q$. There are two natural sources of approximation error, first by treating Q as a fuzzy object and second by approximating the MST weight itself. To model this, we assume that we are given two positive real approximation parameters ε_q and ε_w. Following the typical practice in approximate range searching, the range is expressed as two shapes Q^- and Q^+, where $Q^- \subseteq Q \subseteq Q^+$, and their boundaries are separated by a distance of at least $\varepsilon_q \cdot \mathrm{diam}(Q)$. Points within Q^- must be included and points external to Q^+ cannot be included. A weight W is a valid answer to the query if there exist point sets P' and P'' such that $P \cap Q^- \subseteq P' \subseteq P'' \subseteq P \cap Q^+$ and $wt(\mathrm{MST}(P')) \leq W \leq (1+\varepsilon_w) \cdot wt(\mathrm{MST}(P''))$.

In this paper, we present an efficient data structure for answering such queries. Our approach uses simple data structures based on quadtrees, and it can be applied whenever Q^- and Q^+ are compact sets of constant combinatorial complexity. It uses space $O(n)$, and it answers queries in time $O\big(\log n + 1/(\varepsilon_q\varepsilon_w)^{d+O(1)}\big)$. The $O(1)$ term is a small constant independent of dimension, and the hidden constant factor in the overall running time depends on d, but not on ε_q or ε_w. Preprocessing requires knowledge of ε_w, but not ε_q.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Geometric data structures, Minimum spanning trees, Range searching, Approximation algorithms

Digital Object Identifier 10.4230/LIPIcs.SOCG.2015.781

1 Introduction

Range searching is a fundamental tool in computational geometry. Given a set P of n points in $\mathbb{R}^d$, the objective is to preprocess the points into a data structure so that, given any

* Research supported by the Research Grants Council of Hong Kong, China under project number 16200014.
† Research supported by NSF grant CCF-1117259 and ONR grant N00014-08-1-1015.

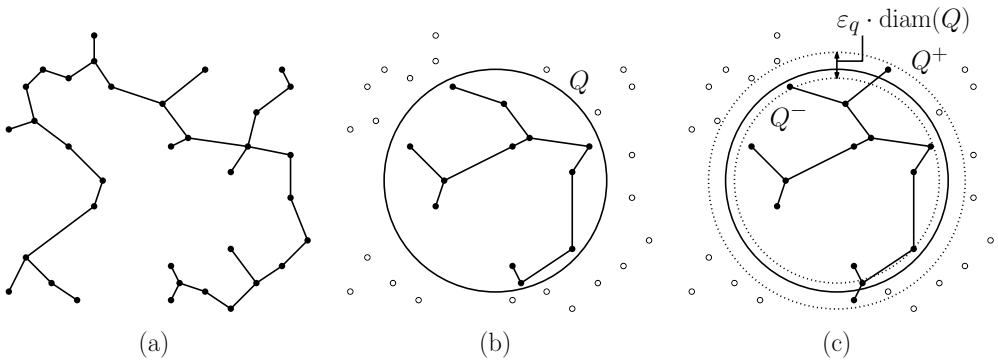

Figure 1 (a) Euclidean MST, (b) MST query, and (c) approximate MST query.

range Q from some class of admissible ranges (e.g., axis-aligned rectangles, balls, halfspaces, simplices), it is possible to efficiently count or report the points of P that lie within Q. Range searching is a powerful method for exploring local regions of a large geometric data set, and it finds many applications in science and engineering.

In many of these applications it is desirable to obtain more detailed information than simple counts. In this paper we explore the question of whether it possible to compute more interesting properties of the subset of points lying within a range, properties that depend on the geometric structure of the points. There are numerous statistics that describe the structure of a point set. Often, such properties are based on graph structures that are implicitly defined by the points set. Perhaps the most fundamental example of such a graph is the Euclidean minimum spanning tree (see Fig. 1(a)). Given a point set P in a Euclidean space, let $\mathrm{MST}(P)$ denote P's minimum weight spanning tree, and let $wt(\mathrm{MST}(P))$ denote its total edge weight. Given a query range Q, an *MST query* returns $wt(\mathrm{MST}(P \cap Q))$ (see Fig. 1(b)). The MST weight (and more generally the distribution of its edge weights) can provide useful information about the density properties of a point set.

Because of the high computational complexities of exact range searching and computing exact geometric spanning trees in multi-dimensional spaces, it is natural to consider the problem in an approximate context. We assume that we are given two positive real parameters ε_q and ε_w, which represent the allowable errors in approximating the query shape and the MST weight, respectively. A range is modeled as a "fuzzy" region of space, so that points near the range's boundary may be included or excluded at the algorithm's discretion. To make this more formal, an *ε_q-approximate range* Q is presented as a pair of compact bodies Q^- and Q^+ (called the *inner range* and *outer range*, respectively), where $Q^- \subseteq Q \subseteq Q^+$ and the boundaries of Q^- and Q^+ are separated by a distance of at least $\varepsilon_q \cdot \mathrm{diam}(Q)$. In standard approximate range searching, the objective is to compute the size (or generally weight) of any set P', such that $P \cap Q^- \subseteq P' \subseteq P \cap Q^+$. Thus, a natural formulation[1] would be to return any weight W such that

$$wt(\mathrm{MST}(P')) \;\leq\; W \;\leq\; (1+\varepsilon_w) \cdot wt(\mathrm{MST}(P')), \text{ where } P \cap Q^- \subseteq P' \subseteq P \cap Q^+.$$

Because we amortize the cost of our result against the weight of the MST in a slightly larger

[1] Note that the "obvious" formulation of returning a weight W such that $wt(\mathrm{MST}(P \cap Q^-)) \leq W \leq (1+\varepsilon_w) \cdot wt(\mathrm{MST}(P \cap Q^+))$ is not well defined because (in dimensions three and higher) there exist point sets such that, even for spherical ranges, $wt(\mathrm{MST}(P \cap Q^-)) > wt(\mathrm{MST}(P \cap Q^+))$. The phenomenon is related to the effect of decreasing the MST weight through the addition of Steiner points.

region, we introduce two sets in our formulation. In particular, we return a weight W such that

$$wt(\mathrm{MST}(P')) \;\le\; W \;\le\; (1+\varepsilon_w)\cdot wt(\mathrm{MST}(P'')), \text{ where } P\cap Q^- \subseteq P' \subseteq P'' \subseteq P\cap Q^+.$$

We refer to this as an $(\varepsilon_q, \varepsilon_w)$-*approximate MST query.*

Our main result is given in the following theorem. For our purposes, a range $Q \subseteq \mathbb{R}^d$ is *admissible* if it is compact and has the property that in $O(1)$ time it is possible to determine for any hypercube b: (1) whether b is contained within Q^+ and (2) whether b is disjoint from Q^-. Thus, the inner and out ranges need not be convex, but should be of constant combinatorial complexity. To simplify the complexity bounds (which are stated in full detail at the end of Section 3.3), we use the notation O^* to ignore factors of the form $1/\varepsilon^{O(1)}$, where the $O(1)$ term does not depend on d (and is roughly 2 in our case).

▶ **Theorem 1.** *Given a set P of n points in $\mathbb{R}^d$ and a weight-approximation parameter $\varepsilon_w > 0$, P can be preprocessed into a data structure of space $O(n)$ such that given any admissible ε_q-approximate query Q, it is possible to answer $(\varepsilon_q, \varepsilon_w)$-approximate MST queries in time $O^*(\log n + 1/(\varepsilon_q\varepsilon_w)^d)$.*

Preprocessing time will be discussed in the full version of the paper, where we show that (ignoring logarithmic factors) the data structure can be built in time $\widetilde{O}(n/\varepsilon^{d/2})$. While preprocessing assumes knowledge of ε_w, it is interesting to note that the space bounds do not depend on ε_w. In [5] it is shown that answering ε_q-approximate range counting queries even for hypercube ranges by searching a partition tree requires $\Omega(\log n + 1/\varepsilon_q^{d-1})$ time. Thus, ignoring the ε_w term, the query time is not far from optimal assuming an approach based on partition trees (as is the approach presented here).

The notion of extracting more complex information than simple counts (or more generally evaluating sums over a commutative semigroup) in range searching has been studied before. One broad class of results involve extensions of *aggregate range searching* [19, 1]. Papadias *et al.* [16] and Shan *et al.* [18] both present data structures that answer various types of nearest neighbor queries over ranges. Nekrich and Smid [15] present a generic data structure that returns an ε-coreset for orthogonal query ranges in $\mathbb{R}^d$. Brass *et al.* [9] present data structures for answering orthogonal range queries in $\mathbb{R}^2$ involving extent measures of the points lying within a query range, including width, area and perimeter of the convex hull, and the smallest enclosing disk. MST queries are particularly challenging because, due to the requirement that the MST must be connected, it is not possible to merely aggregate information in order to answer the query.

Extracting structural information has also been explored in a temporal setting in the work of Bannister *et al.* [8, 7]. In [8] a collection of pairwise relational events are given with time stamps, and it is shown how to extract graph properties efficiently for the events lying within a given query time interval. In [7], this is extended to geometric structures for points with time stamps. Because we are interested in constructing information about the MST in sublinear time, our methods bear similarity to sublinear time algorithms for computing geometric spanning trees, as exemplified in the work of Czumaj, Sohler, and others [12, 13] and Frahling *et al.* [14]. We note, however, that in contrast to these algorithms that are randomized and return only an approximation to the weight (not the edges), our query algorithm is deterministic and implicitly provides a certificate in the form of a connected graph (possibly containing cycles) that spans the point set P' and satisfies the stated weight requirements. Given this certificate, it is possible to enumerate or randomly sample from the edges of this graph.

Our approach borrows some standard techniques for computing approximate geometric spanning trees, such as quadtrees, well-separated pair decompositions (WSPDs), bottom-up construction, and randomized shifting (see, e.g., [11, 4, 2]). Due to the special nature of our problem, we have developed a number of new twists on these ideas. For example, in order to avoid problems with bad quadtree alignments, we develop a local variant of the well-known technique of randomly shifting the coordinate system [3]. We also develop a more efficient method for computing the closest pair of points in the pairs of a WSPD, which exploits the fact that (in our context) the approximation error can be amortized against the weight of the MST within the dumbbell heads of the WSPD.

2 Preliminaries

In this section we provide basic definitions of a number of concepts that will be used throughout the paper.

2.1 Minimum Spanning Trees

Consider a finite point set $P \in \mathbb{R}^d$. Given two points $p, q \in \mathbb{R}^d$, we denote their Euclidean distance by $\|pq\|$. Formally, the *minimum spanning tree* of P, denoted $\mathrm{MST}(P)$, is any minimum spanning tree of the complete graph on P whose edge weights are the interpoint distances. (Our results can be extended easily to any Minkowski distance, with a slight adjustment in the constant factors.) The edges of $\mathrm{MST}(P)$ are line segments, and we will often treat the relevant portions of the MST as a finite set of line segments. Define the *weight* of any such set S of segments, denoted $wt(S)$, to be the sum of the segment lengths.

Throughout, we will need to refer to various restrictions of the edges/weight of the MST to a region of space. We use the term *global MST* to refer to $\mathrm{MST}(P)$. Given subsets $P', P'' \subseteq P$, define the *induced MST* on (P', P''), denoted $\mathrm{MST}(P) \restriction (P', P'')$, to be the subset of global MST edges that have one endpoint in P' and one in P''. Let $\mathrm{MST}(P) \restriction P'$ denote $\mathrm{MST}(P) \restriction (P', P')$.

Given a closed region of space b (which for us will be a hypercube or the difference of two nested hypercubes), there are two natural ways of restricting $\mathrm{MST}(P)$ to b, depending on whether we include edges entirely or partially. Define $\mathrm{MST}(P) \restriction (P \cap b)$ to be the subset of the edges of $\mathrm{MST}(P)$ both of whose endpoints lie within b (see Fig. 2(b)), and define $\mathrm{MST}(P) \restriction b$ to be intersection of $\mathrm{MST}(P)$ (as a set of segments) with b (see Fig. 2(c)). Observe that $\mathrm{MST}(P) \restriction (P \cap b)$ is a subgraph of $\mathrm{MST}(P \cap b)$. When P is understood from context, define the *local connectors* of b, denoted $\Delta(b)$, to be the segments of $\mathrm{MST}(P \cap b)$ that are not in $\mathrm{MST}(P) \restriction (P \cap b)$ (highlighted in Fig. 2(d)).

Our algorithm will classify edges of the MST as being "short" or "long," and process each group differently. Given any $\gamma > 0$, define the *γ-restricted MST*, denoted $\mathrm{MST}_{\gamma}(P)$, to be the subgraph of $\mathrm{MST}(P)$ consisting of edges of weight at most γ, and define $\mathrm{MST}_{>\gamma}(P)$ similarly but for edges of weight greater than γ.

We will organize the edges of the MST using a quadtree decomposition. In general, a uniform grid of hypercubes overlaid on P naturally induces a graph whose vertices are the grid cells and two cells (b, b') are connected by an edge if $\mathrm{MST}(P) \restriction (P \cap b, P \cap b')$ is nonempty. (Note that this graph may contain cycles and self-loop edges.) It is well known that the MST of any finite point set P in $\mathbb{R}^d$ has constant degree (depending on the dimension), and it is easy to show that this is true for this induced graph as well. We omit the proof.

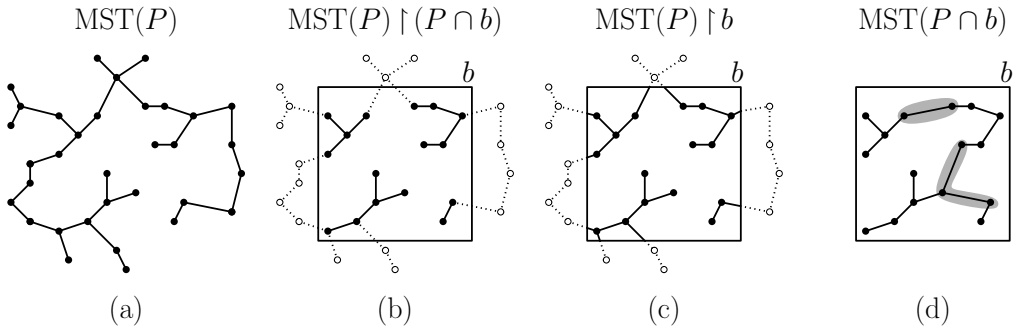

Figure 2 Geometric minimum spanning tree definitions.

▶ **Lemma 2.** *Given a finite point set P in $\mathbb{R}^d$ and a uniform grid of hypercubes, there exists a constant c (depending only on the dimension d) such that the MST induced on the grid is of degree at most c.*

2.2 BBD-trees and Blocks

Our solution will be based on a balanced variant of a quadtree, called a BBD-tree. We refer the reader to [6] for details, but informally, a BBD-tree is based on a quadtree-like subdivision of space, which introduces a decomposition operator, called *shrinking*, that allows the data structure to zoom into regions of dense concentration. The relevant properties of the BBD-tree are given in the following lemma, which was proved by Arya *et al.* [6].

▶ **Lemma 3** (BBD-tree Construction and Packing Lemma)**.** *Given an n-element point set P in $\mathbb{R}^d$, in $O(n \log n)$ time it is possible to construct a BBD-tree of size $O(n)$ and height $O(\log n)$. Furthermore, the number of cells of this tree with pairwise disjoint interiors, each of side length at least s, that intersect a ball of radius r is at most $O((1 + r/s)^d)$.*

For the purposes of processing queries, it will be convenient to conceptualize the subset of points contributing to the query as union of the points lying within a subset of sufficiently small disjoint quadtree boxes all of equal side length. To make this more formal, we introduce the notions of mini-blocks and micro-blocks.

For a sufficiently small constant c (specified later), define $\varepsilon = c \cdot \varepsilon_q \cdot \text{diam}(Q)$. We will assume that c is chosen so that ε is power of two and $c \leq 1/2d$. Define a *mini-block* to be a nonempty quadtree box of side length ε. Let $B_\varepsilon(Q)$ denote the set of mini-blocks that overlap Q^- (the shaded squares of Fig. 3(a)). (This set depends on P and ε_q as well, but since P and ε_q will be fixed throughout, we omit reference to them.) Also, define $B_\varepsilon^+(Q)$ to be the set of quadtree boxes of side length ε such that at least one of its 3^d neighboring blocks is in $B_\varepsilon(Q)$ (all the squares of Fig. 3(a)). A box of side length ε has diameter at most $d\varepsilon \leq (\varepsilon_q/2) \cdot \text{diam}(Q)$, and therefore, all the boxes of $B_\varepsilon(Q)$ and $B_\varepsilon^+(Q)$ lie within Q^+.

An important part of our construction will involve expanding and shifting mini-blocks. Each mini-block b of $B_\varepsilon(Q)$ will be associated with a hypercube that contains b and whose side length is twice as large as b's (see Fig. 3(b)). We call this the *shifted block* and denote it by $\overrightarrow{b}$. Observe that each shifted block lies within the union of the 3^d neighboring blocks of b, and therefore each shifted block lies within $B_\varepsilon^+(Q)$. For a sufficiently small positive constant c' (specified later), define $\delta = c'\varepsilon_w\varepsilon$. Again, we will assume that c' is chosen so that δ is a power of two. Define a *micro-block* (associated with Q) to be a nonempty quadtree box of side length δ. Our preprocessing algorithm will construct $\overrightarrow{b}$ so that it is aligned with the

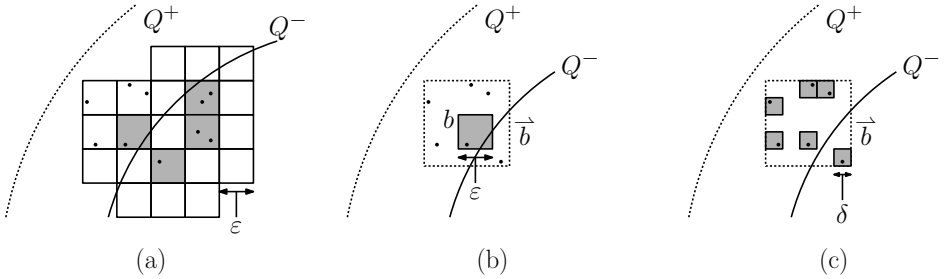

Figure 3 (a) Mini-blocks (all blocks are in $B_\varepsilon^+(Q)$ and shaded blocks are in $B_\varepsilon(Q)$), (b) a mini-block b and its shifted block $\vec{b}$, and (c) the micro-blocks associated with b.

quadtree grid of side length δ. Define $B_\delta(b)$ to be the set of micro-blocks lying within $\vec{b}$ (see Fig. 3(c)), and define $B_\delta(Q)$ to be the union of these micro-blocks over all $b \in B_\varepsilon(Q)$.

Assuming the existence of these quantities for now, define $P(Q)$ to be the subset of P that is covered by all the shifted miniblocks, and similarly define $P^+(Q)$ to be the subset of P lying within the blocks of $B_\varepsilon^+(Q)$. The following results are straightforward consequences of our definitions. (Due to space limitations, proofs have been omitted from this version.)

▶ **Lemma 4.** *There exist constants c and c' (for the above definitions) such that, given a point set P in $\mathbb{R}^d$ and an ε_q-approximate range Q:*

(i) *$B_\varepsilon(Q)$ and $B_\varepsilon^+(Q)$ are both of size $O(1/\varepsilon_q^d)$.*
(ii) *$B_\delta(Q)$ is of size $O(1/(\varepsilon_q\varepsilon_w)^d)$.*
(iii) *$P \cap Q^- \subseteq P(Q) \subseteq P^+(Q) \subseteq P \cap Q^+$.*

This lemma suggests a means by which to construct a solution to an $(\varepsilon_q, \varepsilon_w)$-approximate MST query. Namely, find the weight of the edges of $\mathrm{MST}(P) \restriction P(Q)$, and then include additional edges of low weight to join the connected components of this forest. Our approach will be of this general form, and the additional edges will be classified as being of one of two types, short edges and long edges. At a first reading it is reasonable to think of the sets $P(Q)$ and $P^+(Q)$ as playing the roles of P' and P'' in the definition of an approximate MST query. (But a twist will enter at the end.)

Our next lemma shows that these block sets can be computed efficiently. It is a straightforward adaptation of standard algorithms on BBD-trees.

▶ **Lemma 5.** *Given a BBD-tree storing P and an ε_q-approximate range Q, it is possible to compute $B_\varepsilon(Q)$ and $B_\varepsilon^+(Q)$ in $O(\log n + 1/\varepsilon_q^d)$ time and $B_\delta(Q)$ in $O(\log n + 1/(\varepsilon_q\varepsilon_w)^d)$ time.*

We would like to identify the mini- and micro-blocks with subsets of nodes of the BBD-tree. This is complicated by the fact that a given block need not exist as the cell of any node within the tree because the decomposition ended at a leaf node before reaching this level. In order to focus on the key issues, it will greatly simplify matters to ignore the BBD-tree structure for now and assume that we have instantaneous access to the data stored in any quadtree box. In the full version we will discuss the technical details underlying this assumption.

3 Computing the MST Weight

In this section, we will present our data structure and discuss query processing. Let us begin with a high-level overview of our approach. First, recall that $B_\varepsilon(Q)$ denotes the set of mini-blocks of side length roughly $\varepsilon_q \cdot \mathrm{diam}(Q)$ that overlap the inner query range. These

mini-blocks all lie within the outer query range, and so if we could compute (approximately) the MST of the point set lying within them we would be done. We know that the global MST induced on this set of points is a subset of the final MST. Thus, a natural strategy would be to store the weights of edges of the global MST locally in the nodes of the quadtree, and then at query time combine the MST edge weights for the nodes representing $B_\varepsilon(Q)$ and explicitly compute the additional connecting edges needed to join the connected components of this forest into a single tree.

The difficulty in carrying out this strategy is that there may be many ($\Omega(n)$) connected components of the global MST, and like the tangled branches of a vine, these components can be quite long and intricate and may be separated by arbitrarily small distances. To overcome this problem, within each mini-block we would like to compute (as a part of the preprocessing) a set of edges that will connect the components within this block. Because this will be done independently for each block, without consideration of global connectivity, the problem is determining how to do this without significantly increasing the total edge weight within the query region.

To overcome this problem, we will modify a common strategy used in the computation of geometric MSTs. First, let us focus on "short edges." Recall that δ is roughly $\varepsilon_w\,\varepsilon$, and the δ-restricted MST is the subgraph of the MST(P) consisting of edges of length at most δ. Rather than connecting all the components, we will focus instead on connecting just the components of the δ-restricted MST lying within each mini-block b in order to form the δ-restricted MST of $P \cap b$. (For technical reasons, we will do this for a slightly larger value, $\widehat{\delta} = 2d\delta$, but we will ignore this small variation for now.) Unfortunately, such a local strategy may introduce unnecessarily long edges if the quadtree structure is badly aligned with respect to the point set. In traditional MST approximation algorithms this difficulty is handled by introducing a modified distance function that penalizes very short edges (of length at most δ) that cross the mini-block boundary. This relies on the fact that if a random shift is applied to the coordinate system, then in expectation this added penalty increases the global MST weight by only a small amount. This approach cannot be applied in our setting however, because we need to show that the weight increase is bounded within *every* possible query region.

Rather than shifting the coordinate system, we instead expand each mini-block b by a factor of two and take an appropriately translated copy of this *shifted block*, denoted $\overrightarrow{b}$, that contains b. (For technical reasons, this will be applied to a slight enlargement of the shifted box, called $\overrightarrow{b}^+$.) Because this is computed at preprocessing time, query processing is deterministic. The key property possessed by $\overrightarrow{b}$ is that the total weight needed to connect the δ-restricted global MST within $\overrightarrow{b}$ is within a factor of roughly ε_w of the total weight of the global MST induced in the neighborhood of b, more formally, within the region covered by the 3^d blocks that surround b. We call these additional edges *local connectors*. Recall that $P(Q)$ denotes the union of the points of P lying within these shifted blocks.

Given the weight of the δ-restricted global MST induced on the shifted blocks and the weight of the local connectors, we can now resume our original strategy. We decompose the shifted blocks into micro-blocks of side length δ, accumulate the weights of the δ-restricted global MST and local connectors on these blocks. The number of such blocks is $O(1/(\varepsilon_q\varepsilon_w)^d)$, and this accumulation can be performed within this time bound by a traversal of the BBD-tree. These edges induce a graph on the δ-blocks, called the *global connection graph*. We compute the connected components of this graph. This provides us with an approximation to the δ-restricted MST of $P(Q)$, with the caveat that the approximation error is expressed with respect to the larger point set that lies in $B_\varepsilon^+(Q)$, the neighboring blocks of $B_\varepsilon(Q)$. We refer to all of this as the *short-edge processing*.

To finish the job, we need to add the "long edges" (of length greater than δ) in order to connect the components of the global connection graph. To do this, we employ a strategy based on the well-separated pair decomposition of the micro-blocks. Callahan and Kosaraju [11] observed that, even with a constant factor separation, the MST could be well approximated by computing an approximation to the closest pair within each well-separated pair, and then computing the MST of these pairs. We will apply the same idea with two modifications. Because we are only interested in well-separated pairs at distance greater than $\Theta(\delta)$, the number of pairs is proportional to the number of micro-blocks. Second, we ignore any pairs that join two points whose micro-blocks are within the same component of the global-connection graph.

The problem with applying the Callahan and Kosaraju approach directly is that in order to compute an ε_w-approximation to the closest pair, we would need to decompose each micro-block further into $O\big(1/\varepsilon_w^{\Omega(d)}\big)$ subblocks, which would increase the running time considerably. In order to avoid this additional blow-up, we employ a novel idea. The pairs that are difficult to process are those having many subblocks within the dumbbell head of the well-separated pairs. In such cases, however, the weight of the MST within the dumbbell head is relatively large. Rather than charging the approximation error to the length of the pair returned, we instead charge the error to the weight of the MST within the dumbbell heads. We show that by doing this, the running time is $O((1/\varepsilon_w^2)\log^2(1/\varepsilon_q\varepsilon_w))$, which avoids ε dependencies that grow exponentially in the dimension.

The final answer to the query is the sum of the weights from the short-edge and long-edge processing. As mentioned above, our algorithm is deterministic and implicitly provides a certificate to the answer in the form of a connected graph on $P(Q)$.

3.1 Short-Edge Processing

Let us discuss now the short-edge processing in greater detail. Recall ε, δ, $\widehat{\delta}$, $B_\varepsilon(Q)$, $B_\delta(Q)$, $P(Q)$, and $P^+(Q)$ introduced earlier. Also recall that each mini-block b is associated with a shifted block $\overrightarrow{b}$ (to be specified below), which contains b and is contained within b's neighbors. The objective of this phase is to compute a locally connected augmentation of the $\widehat{\delta}$-restricted global MST within each of the shifted mini-blocks. This will involve three things: (1) the weight of the edges of the $\widehat{\delta}$-restricted global MST induced on each shifted block, (2) the weight of a set of local connectors that join components of this graph to form the $\widehat{\delta}$-restricted MST within each shifted block, and (3) a global-connection graph on the micro-blocks of $B_\delta(Q)$ that connects these components throughout the query range. In this section, we will show that these structures satisfy two properties:

Low weight: The total weight of the local connectors over all the mini-blocks of $B_\varepsilon(Q)$ is at most $(\varepsilon_w/2)\cdot wt(\mathrm{MST}_{\widehat{\delta}}(P^+(Q))$. (This will be established in Lemma 8 below.)

Local connectivity: Given two points $p, p' \in P(Q)$ such that $\|pp'\| \le \widehat{\delta}$, the micro-blocks of $B_\delta(Q)$ that contain these points are in the same connected component of the global-connection graph.

The challenge in achieving these two properties arises from the possible poor placement of partitioning cuts in the quadtree. For example, suppose we have a pair b and b' of neighboring mini-blocks, and we have a large number of point pairs where one element of each pair lies in b and the other in b', and further the segment joining each pair is extremely short (see Fig. 4(a)). If we build the MSTs independently within each mini-block, the local weight will be nearly twice the optimum (see Figs. 4(b) and (c)). Since this instance is the result of an

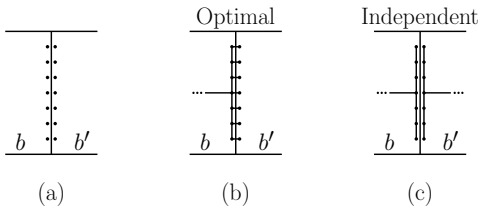

Figure 4 (a) Two points sets lying close to a quadtree splitting edge, (b) the optimal MST, and (c) two MSTs computed independently within each box.

unlucky choice of quadtree cuts, this is usually remedied by applying a random translation to the coordinate system before building the quadtree. While it can be shown that this fixes the problem (in expectation) for the global MST, it does not necessarily fix the problems at the local level, which is what we need for range searching.

As mentioned above, our solution will involve expanding each mini-block by a factor of two, and applying a shift to this expanded block. Before presenting our shifting algorithm, we present a useful lemma. To motivate this lemma, for any $\gamma > 0$ consider the γ-restricted MST of a point set P and a sufficiently large hypercube b. As observed earlier, the γ-restricted global MST induced on $P \cap b$ (formally, $\text{MST}_\gamma(P)\restriction(P \cap b)$) is a subgraph of the γ-restricted MST on $P \cap b$ (formally, $\text{MST}_\gamma(P \cap b)$). Define $\Delta_\gamma(b)$ to be the edges in the set-theoretic difference of these two graphs. We will show that $wt(\Delta_\gamma(b))$ is proportional to the weight of the global spanning tree within distance γ of b's boundary. Intuitively, this holds because the components of $\text{MST}_\gamma(P)\restriction(P \cap b)$ that are connected in $\text{MST}_\gamma(P \cap b)$ must be connected by paths consisting of edges of the MST of length at most γ that lie outside of b.

Before stating the lemma we introduce some terminology. Given a hypercube b of side length at least 2γ, define the *γ-shell* of b, denoted $\text{shell}_\gamma(b)$, to be the set-theoretic difference of two hypercubes b^+ and b^-, where $b^- \subset b \subset b^+$, and the boundaries of these hypercubes are separated from b's boundary by a distance of γ.

▶ **Lemma 6.** *Consider a point set P in $\mathbb{R}^d$, $\gamma > 0$, and a hypercube b of side length at least 2γ. Then, $wt(\Delta_\gamma(b)) \leq 3 \cdot wt(\text{MST}_\gamma(P)\restriction\text{shell}_\gamma(b))$.*

Resuming the discussion of the short-edge processing, consider a mini-block b. Recall that its side length is ε. For the sake of our construction, let us assume that the origin is centered at b's center. Let $2b$ and $3b$ denote centrally scaled copies of b by factors of 2 and 3, respectively (see Fig. 5). Because we are interested in edges of length up to $\widehat{\delta}$ that might have one endpoint within $2b$ and one endpoint outside, let $2b^+$ denote the hypercube that results by translating each of the bounding hyperplanes of $2b$ outwards by distance $\widehat{\delta}$. Given a vector $\mathbf{v}$ let $2b^+ + \mathbf{v}$ denote a translation of $2b^+$ by $\mathbf{v}$.

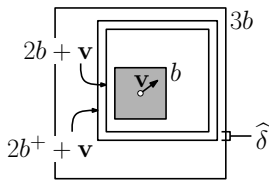

Figure 5 An expanded and shifted block.

Recalling the definitions of Section 2.2, our objective is to compute the shifted block $\overrightarrow{b}$ to be associated with b. To do so, we will consider a set of $O((\varepsilon/\delta)^d)$ possible shifts of $2b^+$, each of which will contain b and lie within $3b$. Our next lemma shows that for at least one of these shifts (in fact, for a constant fraction of them) the local connection weight $wt(\Delta_{\widehat{\delta}}(2b^+ + \mathbf{v}))$ is $O(\varepsilon_w)$ times the weight of the $\widehat{\delta}$-restricted MST induced on $P \cap 3b$.

▶ **Lemma 7.** *Consider a point set P in $\mathbb{R}^d$, an approximate query Q, and a mini-block $b \in B_\varepsilon(Q)$. For any constant $c'' > 0$, there exists a translate of $2b^+$, denoted $\widehat{b}$, that is nested between b and $3b$, is aligned with the quadtree grid of side length δ, and*

$$wt\big(\Delta_{\widehat{\delta}}(\widehat{b})\big) \;\leq\; c'' \cdot \varepsilon_w \cdot wt(\text{MST}_{\widehat{\delta}}(P)\restriction(P \cap 3b)).$$

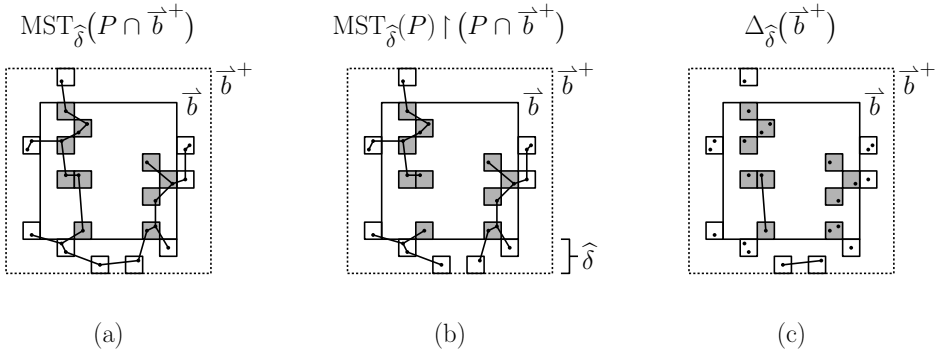

Figure 6 A shifted mini-block $\overrightarrow{b}$ and its expansion $\overrightarrow{b}^{+}$. The micro-blocks $\mu(\overrightarrow{b})$ are shown as shaded boxes and $\mu(\overrightarrow{b}^{+})$ includes the white boxes as well. (a) The restricted MST of $P \cap \overrightarrow{b}^{+}$, (b) the induced global spanning tree on $P \cap \overrightarrow{b}^{+}$, and (c) the local connectors.

Given a mini-block b, define $\overrightarrow{b}^{+}$ to be the translated box $\widehat{b}$ from the above lemma, and define its shifted block, $\overrightarrow{b}$, to be the corresponding translate of $2b$. This information is computed for each node of the quadtree as part of the preprocessing.

To complete the short-edge processing, we need to compute the local connectors (that is, the edges of the $\widehat{\delta}$-restricted MST on $P \cap \overrightarrow{b}$ that are not in the $\widehat{\delta}$-restricted MST induced on these points). An obvious approach would be to compute $\mathrm{MST}_{\widehat{\delta}}(P \cap \overrightarrow{b})$, and then remove from this the edges from the global MST. While this would work fine for an individual shifted block, this is not sufficient to guarantee connectivity across the entire query region (particularly for blocks near the query's boundary). Since the edges involved are all of length at most $\widehat{\delta}$, for the purposes of computing connectivity, we will consider micro-blocks that lie slightly (distance at most $\widehat{\delta}$) outside the shifted blocks. Once the connected components have been computed, we will discard these extra blocks.

To make this more precise, given a mini-block b, define $\mu(\overrightarrow{b})$ to be the micro-blocks that lie within $\overrightarrow{b}$ (the shaded small boxes in Fig. 6), and define $\mu(\overrightarrow{b}^{+})$ similarly for $\overrightarrow{b}^{+}$ (all the small boxes in Fig. 6). To compute the local connectors, at preprocessing time for each such mini-block b, we compute $\overrightarrow{b}^{+}$ (by the previous lemma) and $\mathrm{MST}_{\widehat{\delta}}(P \cap \overrightarrow{b}^{+})$ (see Fig. 6(a)). We assume that the global MST has already been computed. The local connectors consist of the edges that are not already in the global spanning tree induced on these points, that is,

$$\Delta_{\widehat{\delta}}(\overrightarrow{b}^{+}) \;=\; \mathrm{MST}_{\widehat{\delta}}(P \cap \overrightarrow{b}^{+}) \setminus \mathrm{MST}_{\widehat{\delta}}(P) \restriction (P \cap \overrightarrow{b}^{+})$$

(see Figs. 6(b) and (c)).

We cannot deal with structures like $\Delta_{\widehat{\delta}}(\overrightarrow{b}^{+})$ at query time, since they involve individual points. Instead, we will deal with graphs that they induce on the micro-blocks. At preprocessing time, we compute an induced (weighted) graph on the micro-blocks of $\mu(\overrightarrow{b}^{+})$ from the local connectors as follows. For each edge (p, p') in $\Delta_{\widehat{\delta}}(\overrightarrow{b}^{+})$, create an edge between the respective micro-blocks b and b' that contain them. Set the weight of this edge to be the total length of all such edges. Because each edge of $\Delta_{\widehat{\delta}}(\overrightarrow{b}^{+})$ is of length at most $\widehat{\delta}$, the neighbors of each micro-block (whose side length is δ) lie within distance at most $\widehat{\delta}$. The number of such neighbors is $O((\widehat{\delta}/\delta)^d) = O(d^d) = O(1)$. Therefore, this graph has constant degree. Also, as a part of preprocessing, we compute the weight of the edges of the $\widehat{\delta}$-restricted global MST induced on each pair of micro-blocks. (This is done implicitly. See the full version for details.)

When processing a query Q, we combine the aforementioned graphs at the mini-block level to derive two additional global structures on the micro-block level. The first is a structure that encapsulates all the local-connector weight at the micro-block level. Define $\Delta_{\widehat{\delta}}(Q)$ to be the union of the graphs $\Delta_{\widehat{\delta}}(\overrightarrow{b}^+)$ over all mini-blocks $b \in B_\varepsilon(Q)$. This is a structure on points, but we can compute its micro-block induced structure by taking the union of the corresponding micro-block structures mentioned above. If there is an edge (b, b') between the same pair of micro-blocks appearing in multiple shifted blocks (which can happen if their shifted blocks overlap), then assign the edge weight to be the sum over all the contributing edges. (We do this because each edge reflects potentially different pairs of locally connected points, and we need to account for the entire weight of these connections. Note that because these involve expanded shifted blocks ($\overrightarrow{b}^+$), this will implicitly count the weight of edges whose endpoints lie within $P^+(Q)$ but not $P(Q)$.) The total edge weight of this induced graph is the same as $\Delta_{\widehat{\delta}}(Q)$.

Our next lemma bounds the weight of this graph in terms of the weight of the $\widehat{\delta}$-restricted MST of a subset of points lying within the outer query range.

▶ **Lemma 8.** *The weight of $\Delta_{\widehat{\delta}}(Q)$ is at most $(\varepsilon_w/2) \cdot wt(\mathrm{MST}_{\widehat{\delta}}(P^+(Q))$.*

The second structure built at query time is the global-connection graph. It consists of the union of the edges of $\Delta_{\widehat{\delta}}(Q)$ together with the edges of the $\widehat{\delta}$-restricted global MST induced on the points lying within the union of the expansions of the shifted blocks (that is, the union of $\mathrm{MST}_{\widehat{\delta}}(P)\restriction(P \cap \overrightarrow{b}^+)$ over all miniblocks b). As with these other graphs, it is defined on points, but it will be represented as an induced graph on micro-blocks. Since this graph is used only for computing connected components, we do not need to assign weights to its edges.

Summarizing the short-edge processing, the data structure consists of the BBD-tree storing the point set P. Each mini-block b is associated with its shifted block $\overrightarrow{b}$ (and implicitly its expansion $\overrightarrow{b}^+$) and the graph of local connectors (from $\Delta_{\widehat{\delta}}(\overrightarrow{b}^+)$) induced on the micro-blocks of $\mu(\overrightarrow{b}^+)$. We also store the edges of the global MST so that we can efficiently extract the weight of the $\widehat{\delta}$-restricted MST induced on the micro-blocks (details in the full version). The total space is dominated by the size of the BBD-tree, the storage of the edges of the MST, and the storage of the edges of the local connectors, which is $O(n)$.

Details regarding how these structures are used in the query processing are deferred to the full version. The following lemma summarizes the short-edge phase.

▶ **Lemma 9** (Short-edge summary)**.** *Given an n-element point set P in $\mathbb{R}^d$ and an approximation parameter ε_w, there exists a data structure of space $O(n)$ such that given any ε_q-approximate query Q, in time $O(\log n + 1/(\varepsilon_q\varepsilon_w)^d)$ it is possible to compute (implicitly) point sets $P(Q)$ and $P^+(Q)$, a graph $G_s = (P(Q), E_s)$ (which may contain cycles), and a labeling of the connected components of G_s, such that*

- **(i)** *$Q^- \subseteq P(Q) \subseteq P^+(Q) \subseteq Q^+$,*
- **(ii)** *any two points of $P(Q)$ that are within distance $\widehat{\delta}$ of each other lie in the same connected component of G_s, and*
- **(iii)** *the weight of the edges in E_s is at most $wt(\mathrm{MST}_{\widehat{\delta}}(P(Q))) + (\varepsilon_w/2) \cdot wt(\mathrm{MST}_{\widehat{\delta}}(P^+(Q)))$.*

These point sets are represented implicitly by $O(1/(\varepsilon_q\varepsilon_w)^d)$ micro-blocks. The graph G_s is of constant degree, and so is of the same asymptotic size.

3.2 Long-Edge Processing

Given the information from the short-edge processing, as summarized in Lemma 9, let us now consider the long-edge case. Let Ψ denote a well-separated pair decomposition (WSPD) for the point set $P(Q)$ for some suitable constant separation factor (for definitions see [10]). In particular, we require that if (A, A') is a pair of the WSPD, then for all $p, q \in A$ and all $p', q' \in A'$, $\|pp'\| > \max(\|qp\|, \|q'p'\|)$. Drawing on a standard visual analogy, we think of the WSPD as consisting of a collection of *dumbbells*, where each of the sets being separated lies within one of the two *heads* of a dumbbell. Observe that each well-separated pair contributes at most one edge to $\mathrm{MST}(P(Q))$ (because all the points within a dumbbell head will be connected by Kruskal's algorithm before considering any edge between the heads).

Let $\Psi' \subseteq \Psi$ denote the set of dumbbells such that for any pair of points $p, p' \in P(Q)$, where $\|pp'\| > \widehat{\delta}$, there is a dumbbell in Ψ' that separates p and p'. By standard techniques, we can compute Ψ' in time proportional to the number of δ-blocks that cover the points of $P(Q)$, which is $O(1/\delta^d) = O(1/(\varepsilon_q \varepsilon_w)^d)$ (see, e.g., [17]). We assume that every internal node of the BBD-tree contains an arbitrary *representative point* drawn from the points lying within the node's outer box.

Our objective is to compute a suitable approximation to the closest pair of points separated by each dumbbell. Recall from the high-level overview of Section 3 that the classical approach for doing this would involve decomposing each of the dumbbell heads into sufficiently small blocks so that the error committed can be charged against the resulting edge of the MST. Unfortunately, this will result in an unacceptably high running time. In contrast, our approach is sensitive to the weight of the MST in the vicinity of the dumbbell heads. We decompose the blocks in a breadth-first manner until the number of nonempty subblocks in either of the dumbbell heads is roughly $1/\varepsilon_w$. We will exploit the fact that the existence of this many nonempty subblocks implies that the weight of the MST within this dumbbell heads will be sufficient to pay for the approximation error.

More formally, we introduce a parameter α (whose exact value will be specified later but can be thought of as being roughly ε_w). We will process each dumbbell $\psi \in \Psi'$ and compute an edge e_ψ joining a representative point in each head of ψ. We do this as follows. We decompose the two heads of ψ in parallel, always maintaining boxes of equal side length until reaching a total of $\Theta(1/\alpha)$ nonempty quadtree boxes or encountering all the points within the head (whichever occurs first). We then examine the representative points from each pair of boxes and keep the closest pair. This takes time $O(1/\alpha^2)$ by brute-force. We choose an arbitrary point from each box in this pair. The edge joining these two points is selected as the representative edge e_ψ. Let E_s denote the edges of the short-edge graph G_s, and let E_ℓ denote the edges computed above. Let G denote the graph $(P(Q), E_s \cup E_\ell)$.

Just as we did for G_s, we can associate each edge of E_ℓ with the pair of micro-blocks that contain the edge's respective endpoints. This defines a graph on the micro-blocks. To complete the long-edge phase, we first prune this graph. If any edge of this graph joins two micro-blocks in the same short-edge connected component, we ignore this edge. We then collapse all the micro-blocks belonging to the same short-edge component into a single vertex, forming a component graph. For any two components, we keep only the shortest edge between them. Since the number of well-separated pairs is $O(1/(\varepsilon_q \varepsilon_w)^d)$, the number of vertices and edges in this graph is similarly bounded. We then compute the MST of this component graph, using any standard MST algorithm in time $O(1/(\varepsilon_q \varepsilon_w)^d \log 1/(\varepsilon_q \varepsilon_w))$. The output of the long-edge phase is the weight of the edges of E_ℓ that remain in the final MST.

Rather than analyzing G directly, it will be easier to analyze a related graph. Let G' denote the subgraph of G with the same vertex set and the following edges. We keep all the edges of E_s, but only a subset E'_ℓ of the edges of E_ℓ selected as follows. For each edge e of $\mathrm{MST}_{>\widehat{\delta}}(P(Q))$, we select the representative edge e_ψ associated with the dumbbell $\psi \in \Psi'$ that separates the endpoints of e.

In the rest of this section, we will show that G' is connected and satisfies the desired weight bound. Due to space limitations, we will only present the main lemmas upon which the result is based. Details can be found in the full version. Our analysis will employ the following lemma, which bounds the weight of the MST in terms of the number of quadtree boxes (see, e.g., [12]).

▶ **Lemma 10.** *Given a finite point set $P \in \mathbb{R}^d$ and a hypercube grid of side length s, let $m(P)$ denote the number of cells of the grid that contain a point of P. Then $wt(\mathrm{MST}(P)) \geq (s/2) \cdot ((m(P)/2^d) - 1)$.*

For any dumbbell $\psi \in \Psi'$, define z_ψ to be the distance between the closest pair of points that are separated by ψ. The following lemma bounds the total error incurred in selecting the long edges.

▶ **Lemma 11.** *There exists a constant c (depending on dimension) such that*

$$\sum_{\psi \in \Psi'} (wt(e_\psi) - z_\psi) \;\leq\; c \cdot \alpha \cdot \log(1/(\varepsilon_q \varepsilon_w)) \cdot wt(\mathrm{MST}(P(Q))).$$

Setting $\alpha = \varepsilon_w/(4c \cdot \lg(1/(\varepsilon_q \varepsilon_w)))$, by the above lemma, the long edges satisfy the following property:

$$\sum_{\psi \in \Psi'} (wt(e_\psi) - z_\psi) \;\leq\; \frac{\varepsilon_w}{4} \cdot wt(\mathrm{MST}(P(Q))). \tag{1}$$

The connectedness of G' follows from the WSPD separation properties.

By combining Eq. (1) above with our earlier observation that each dumbbell contributes at most one edge to $\mathrm{MST}(P(Q))$, it follows that the weight of the long edges of G', namely $wt(E'_\ell)$, is at most $wt(\mathrm{MST}_{>\widehat{\delta}}(P(Q))) + (\varepsilon_w/4) \cdot wt(\mathrm{MST}(P(Q)))$. Because G' connects the components of E_s, its weight cannot be smaller than the MST weight of the component graph, which is the output of this phase. Therefore, we have the following.

▶ **Lemma 12** (Long-edge summary)**.** *Given the output from the short-edge processing, in time $O((1/(\varepsilon_q^d \varepsilon_w^{d+2})) \log^2(1/\varepsilon_q \varepsilon_w))$, we can output a set of edges that connects all the short-edge components and whose total weight is at most $wt(\mathrm{MST}_{>\widehat{\delta}}(P(Q))) + (\varepsilon_w/4) \cdot wt(\mathrm{MST}(P(Q)))$.*

3.3 Combining the Short and Long Edges

Let us now combine the results of the short-edge and long-edge phases. By Lemma 9(iii), the total weight of the short edges E_s is at most

$$wt(\mathrm{MST}_{\widehat{\delta}}(P(Q))) + \frac{\varepsilon_w}{2} \cdot wt(\mathrm{MST}(P^+(Q)))).$$

By Lemma 12, the total weight of the long-edge phase is at most

$$wt(\mathrm{MST}_{>\widehat{\delta}}(P(Q))) + \frac{\varepsilon_w}{4} \cdot wt(\mathrm{MST}(P(Q))).$$

Combining the weights of both phases, we find that the total weight $W(Q)$ output is at most

$$W(Q) \quad = \quad wt(\text{MST}(P(Q))) + \frac{\varepsilon_w}{2} \cdot wt(\text{MST}(P^+(Q))) + \frac{\varepsilon_w}{4} \cdot wt(\text{MST}(P(Q))).$$

Since $P(Q) \subseteq P^+(Q)$, we have $wt(\text{MST}(P(Q))) \leq 2 \cdot wt(\text{MST}(P^+(Q)))$. (This follows from the facts that Steiner tree weight increases monotonically as points are added and that the weight of the MST is at most twice the weight of the Steiner tree.) Therefore, we have

$$W(Q) \;\leq\; wt(\text{MST}(P(Q))) + \varepsilon_w \cdot wt(\text{MST}(P^+(Q))).$$

If $wt(\text{MST}(P(Q))) \leq wt(\text{MST}(P^+(Q))$, then $W(Q)$ can be bounded by $(1+\varepsilon_w) \cdot wt(\text{MST}(P^+(Q))))$. On the other hand, if $wt(\text{MST}(P(Q))) > wt(\text{MST}(P^+(Q))$, this can be bounded by $(1+\varepsilon_w) \cdot wt(\text{MST}(P(Q))))$. By defining $P' = P(Q)$ and P'' to be whichever set yields the larger MST weight, we obtain the following bound

$$wt(\text{MST}(P')) \;\leq\; W(Q) \;\leq\; (1+\varepsilon_w) \cdot wt(\text{MST}(P'')),$$

where $P \cap Q^- \subseteq P' \subseteq P'' \subseteq P \cap Q^+$. Therefore, this is a valid answer to the $(\varepsilon_q, \varepsilon_w)$-approximate MST query.

By Lemma 9, the running time of the short-edge phase is $O(\log n + 1/(\varepsilon_q\varepsilon_w)^d)$, and by Lemma 12, the running time of the long-edge phase is $O((1/(\varepsilon_q^d \varepsilon_w^{d+2}))\log^2(1/\varepsilon_q\varepsilon_w))$. Thus, the overall query time is $O(\log n + (1/(\varepsilon_q^d\varepsilon_w^{d+2}))\log^2(1/\varepsilon_q\varepsilon_w))$. In summary, we have the following result, which is stated more concisely in Theorem 1.

▶ **Theorem 13.** *Given a set P of n points in $\mathbb{R}^d$ and a weight-approximation parameter $\varepsilon_w > 0$, it is possible to preprocess P into a data structure of space $O(n)$ such that given any ε_q-approximate query Q, it is possible to answer $(\varepsilon_q, \varepsilon_w)$-approximate MST queries in time $O(\log n + (1/(\varepsilon_q^d\varepsilon_w^{d+2}))\log^2(1/\varepsilon_q\varepsilon_w))$.*

4 Conclusions

We have demonstrated an efficient data structure for answering approximate MST range queries. Although our query processing focused only on returning the approximate weight, our data structure implicitly provides much more information. In particular, the weight returned is a accumulation of three disjoint edge sets, the global MST edges induced on the approximate query range, a set of local connecting edges, and the long edges. All of these edges (not just their weights) are stored within the data structure. Thus, unlike sublinear time algorithms for the MST, which provide just an approximation to the weight, our data structure implicitly provides a certificate for its answer. This certificate could be output, which would result in a data structure for approximate MST range reporting queries. Alternatively, the edges of the certificate could be randomly sampled, which would allow a user to compute statistics about this graph, such as the distribution of its edge weights.

There are two obvious shortcomings with our approach. First, our answer is the weight of a graph on a set of points within the approximate query region, which spans these points but may contain cycles. An obvious open problem is whether it is possible to efficiently compute the exact weight of a graph that is a spanning tree on some subset of points that constitutes a valid answer to the approximate range query. Second, our approximation bounds involve two sets P' and P'', one for the lower bound and one for the upper bound. It would be nice to relate the result to the weight of the MST on a single point set.

References

1. P. K. Agarwal, L. Arge, S. Govindarajan, J. Yanga, and K. Yi. Efficient external memory structures for range-aggregate queries. *Comput. Geom. Theory Appl.*, 46:358–370, 2013.
2. A. Andoni, A. Nikolov, K. Onak, and G. Yaroslavtsev. Parallel algorithms for geometric graph problems. In *Proc. 46th Annu. ACM Sympos. Theory Comput.*, pages 574–583, 2014.
3. S. Arora. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *J. Assoc. Comput. Mach.*, 45:753–782, 1998.
4. 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.
5. S. Arya and D. M. Mount. Approximate range searching. *Comput. Geom. Theory Appl.*, 17:135–163, 2000.
6. S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Wu. An optimal algorithm for approximate nearest neighbor searching. *J. Assoc. Comput. Mach.*, 45:891–923, 1998.
7. M. J. Bannister, W. E. Devanny, M. T. Goodrich, J. A. Simons, and L. Trott. Windows into geometric events: Data structures for time-windowed querying of temporal point sets. In *Proc. 26th Canad. Conf. Comput. Geom.*, 2014.
8. M. J. Bannister, C. DuBois, D. Eppstein, and P. Smyth. Windows into relational events: data structures for contiguous subsequences of edges. In *Proc. 24th Annu. ACM-SIAM Sympos. Discrete Algorithms*, pages 856–864, 2013. (arXiv:1209.5791).
9. P. Brass, C. Knauer, C.-S. Shin, M. Smid, and I. Vigan. Range-aggregate queries for geometric extent problems. In *Proc. 19th Computing: The Australasian Theory Symposium (CATS)*, pages 3–10, 2013.
10. 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.
11. P. B. Callahan and S. R. Kosaraju. Faster algorithms for some geometric graph problems in higher dimensions. In *Proc. Eighth Annu. ACM-SIAM Sympos. Discrete Algorithms*, pages 291–300, 1997.
12. 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.
13. A. Czumaj and C. Sohler. Estimating the weight of metric minimum spanning trees in sublinear time. *SIAM J. Comput.*, 39:904–922, 2009.
14. G. Frahling, P. Indyk, and C. Sohler. Sampling in dynamic data streams and applications. *Internat. J. Comput. Geom. Appl.*, 18:3–28, 2008.
15. Y. Nekrich and M. Smid. Approximating range-aggregate queries using coresets. In *Proc. 22nd Canad. Conf. Comput. Geom.*, pages 253–256, 2010.
16. D. Papadias, Y. Tao, K. Mouratidis, and K. Hui. Aggregate nearest neighbor queries in spatial databases. *ACM Transactions on Database Systems (TODS)*, 30:529–576, 2005.
17. E. Park and D. M. Mount. Output-sensitive well-separated pair decompositions for dynamic point sets. In *Proc. 21st Internat. Conf. on Advances in Geographic Information Systems*, pages 364–373, 2013. (doi: 10.1145/2525314.2525364).
18. J. Shan, D. Zhang, and B. Salzberg. On spatial-range closest-pair query. In T. Hadzilacos, Y. Manolopoulos, J. Roddick, and Y. Theodoridis, editors, *Advances in Spatial and Temporal Databases*, volume 2750 of *Lecture Notes in Computer Science*, pages 252–269. Springer, Berlin, 2003.
19. Y. Tao and D. Papadias. Range aggregate processing in spatial databases. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 16:1555–1570, 2004.