Notes on Binary Dumbbell Trees

Michiel Smid*

March 23, 2012

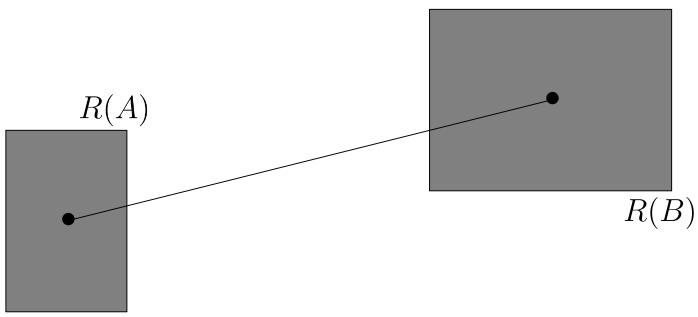
Abstract

Dumbbell trees were introduced in [1]. A detailed description of non-binary dumbbell trees appears in Chapter 11 of [3]. These notes show how binary dumbbell trees can be obtained, and how they can be used to construct, in $O(n \log n)$ time, a spanner of bounded degree and weight proportional to $O(\log n)$ times the weight of a minimum spanning tree. The reader is assumed to be familiar with the split tree and the well-separated pair decomposition (WSPD), as described, e.g., in Chapter 9 of [3].

1 Dumbbells

Let S be a set of n points in $\mathbb{R}^d$. For any subset A of S, we denote its bounding box by $R(A)$. Let T be the split tree for S. For any node u in T, let S_u be the set of all points in S that are stored in the subtree of u. Consider the WSPD that is obtained from T. Each of the $O(n)$ pairs $\{A, B\}$ in this WSPD is represented by two nodes, say u and v, of T: The set A is equal to the set S_u, whereas the set B is equal to the set S_v.

For each pair $\{A, B\}$ in the WSPD, we define its *dumbbell* to be the geometric region consisting of the two bounding boxes $R(A)$ and $R(B)$, together with the line segment joining the centers of these boxes. The two bounding boxes are called the *heads* of the dumbbell. The *length* of the dumbbell is defined to be the length of the line segment joining the centers of the two heads. The *size* of a head is defined to be the length of its longest side.



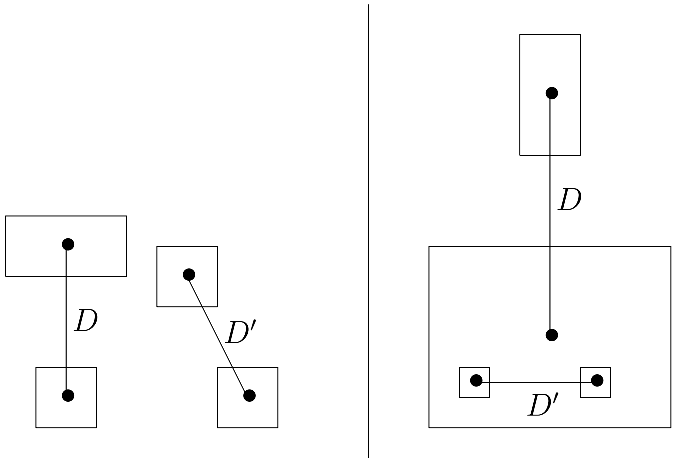
*School of Computer Science, Carleton University, Ottawa, Ontario, Canada.

We add a *dummy dumbbell*, one of whose heads is so large that it contains all other dumbbells. For example, we can take this head to be the bounding box of all points of S.

2 Dumbbell trees for perfectly nested dumbbells

In this section, we present the main idea of the dumbbell trees for the special case when the dumbbells can be partitioned into $O(1)$ groups, such that the dumbbells within each group are "perfectly nested": Assume that for any two distinct dumbbells D and D' in the same group,

- either any head of D is disjoint from any head of D' (see the left figure below),
- or one dumbbell, say D', is completely contained in one head of the other dumbbell D (see the right figure below).



Consider one group in this partition. The dumbbells in this group are *nested* in a natural way: For any non-dummy dumbbell D', there is a unique dumbbell D, such that

- D' is contained in one of the heads, say H, of D,
- and there is no dumbbell D'' such that (i) D' is in one of the heads of D'' and (ii) D'' is in the head H of D.

We say that D' is *nested immediately within* H.

This nesting leads to a *dumbbell tree* for the group in the partition under consideration. This tree consists of *dumbbell nodes*, *head nodes*, and *leaves*:

- The root is a dumbbell node and stores the dummy dumbbell.
- Each dumbbell node (storing, say, dumbbell D) has two children, which are head nodes storing the heads of D.

- Each head node (storing, say, head H) has dumbbell nodes as its children: There is one dumbbell node for each dumbbell that is nested immediately within H. The head node may also have leaves as children; see the next item.

- Each point p of S is stored at one leaf. The parent of this leaf is the deepest head node whose head contains p.

- With each node u in this dumbbell tree, we store a *representative* point $rep(u)$ of S:
  - If u is a leaf, then $rep(u)$ is the point stored at u.
  - If u is not a leaf, then $rep(u)$ is an arbitrary point stored at one of the leaves in the subtree of u.

Thus, for each of the $O(1)$ groups in the partition of the dumbbells, we obtain one dumbbell tree. Define the graph $G = (S, E)$, where

$$E = \{\{rep(u), rep(v)\} : \text{ } u \text{ and } v \text{ are nodes in the same dumbbell tree, } u \text{ is a child of } v\}.$$

We claim that G is a spanner. To prove this, let p and q be two distinct points of S, let $\{A, B\}$ be the pair in the WSPD such that $p \in A$ and $q \in B$, and let D be the dumbbell corresponding to this pair. Consider the dumbbell tree that stores D, and let u be the dumbbell node in this tree that stores D. The left child u_ℓ of u is a head node storing $R(A)$, whereas the right child u_r of u is a head node storing $R(B)$. Let v and w be the leaves storing p and q, respectively. Observe that v is in the subtree of u_ℓ, and w is in the subtree of u_r.

In the dumbbell tree, walk from v to w, and let P be the sequence of representatives encountered. Then P is a path in G from p to q, and it is the concatenation of the following:

- A path starting at $rep(v) = p$ and ending at $rep(u_\ell)$.

  When walking down the dumbbell tree, the lengths of the dumbbells and the sizes of the heads decrease exponentially. As a result, the length of this path is proportional to the size of the head $R(A)$.

- The path $rep(u_\ell)$, $rep(u)$, $rep(u_r)$.

  Since both p and $rep(u_\ell)$ are in A, both q and $rep(u_r)$ are in B, and $rep(u)$ is in A or in B, the length of this path is equal to $|pq|$ plus a term that is at most proportional to the sum of the sizes of the heads $R(A)$ and $R(B)$.

- A path starting at $rep(u_r)$ and ending at $rep(w) = q$.

  The length of this path is proportional to the size of the head $R(B)$.

By choosing the separation ratio of the WSPD sufficiently large, the length of the entire path P is at most $(1 + \epsilon)|pq|$. Thus, G is indeed a spanner for the point set S.

All of this assumed that the dumbbells can be partitioned into $O(1)$ groups, such that the dumbbells within each group are perfectly nested. Unfortunately, this is, in general, not the case:

Exercise 1 Let n be a large integer, let $s > 0$ be a real number (that will be used as the separation ratio), and let $\epsilon > 0$ be a real number such that

$$\epsilon < \frac{s}{2s + 2^{n-2}}.$$

For any integer $i \geq 0$, define

$$p_i = 1/2 + 1/2^{i+1} + \left(1 - 1/2^i\right) \epsilon.$$

Let $S \subseteq \mathbb{R}$ be the set consisting of the n real numbers

$$0, 1/2, 1/2 + \epsilon, p_{n-3}, \ldots p_2, p_1, p_0.$$

Since $\epsilon < 1/2$, the elements in this sequence are sorted. Observe that $p_0 = 1$ and, for each i with $0 \leq i \leq n - 2$, p_{i+1} is the midpoint of $1/2 + \epsilon$ and p_i.

Compute the split tree of S and the corresponding WSPD using the algorithm as described in Chapter 9 of [3]. Assume that, when splitting the current bounding box (which is an interval) in the middle, in case this middle value is an element of S, this element belongs to the left subset of the two resulting subsets.

Show that this WSPD contains the pairs

$$\{\{1/2\}, \{p_0\}\}, \{\{1/2\}, \{p_1\}\}, \ldots, \{\{1/2\}, \{p_{n-3}\}\}, \{\{1/2\}, \{1/2 + \epsilon\}\}.$$

Thus, this WSPD gives $\Omega(n)$ dumbbells that all share one head.

In the next section, we will see that the dumbbells can be partitioned into $O(1)$ groups, such that the dumbbells within each group are "almost" perfectly nested. As a result of this, we get a collection of $O(1)$ dumbbell trees that can be used to define a spanner.

Before we proceed, observe that the dumbbell tree as defined above is not a binary tree, because a head node may have many dumbbell nodes and leaves as children. Also, the spanner G defined above may have a very large weight, as compared to the minimum spanning tree of S: Assume that the dimension d is equal to 2, and consider a large head H of some dumbbell that has a point p close to its top-right corner. Assume that p is the representative of the head node storing H. Also assume that there are millions of tiny and pairwise disjoint dumbbells near the bottom-left corner of H. Each of these tiny dumbbells is a child of H and defines an edge in the spanner; the weight of this edge is close to the diameter, say L, of H. Thus, the spanner G has millions of edges of length close to L. However, the minimum spanning tree of the points involved has only one edge whose length is close to L.

3 A first version of the dumbbell trees

Consider again the dumbbells defined by the pairs in the WSPD. Recall that the length of a dumbbell is the distance between the centers of its two heads. In Sections 11.4 and 11.5 of [3], it is shown that the dumbbells can be partitioned into $O(1)$ groups such that the following two properties hold for any two distinct dumbbells D and D'. Let ℓ and ℓ' be the lengths of D and D', respectively, and assume that $\ell \leq \ell'$.

Length-grouping property: If D and D' are in the same group, then

- either $\ell' \leq 2\ell$, i.e., D and D' have approximately the same lengths,
- or $\ell' \geq \ell/\beta$, i.e., D' is much longer than D. Here, β is a small positive real constant.

Empty-region property: If D and D' are in the same group and $\ell' \leq 2\ell$, then the distance between any head of D and any head of D' is at least $\gamma\ell$, where γ is a large constant. Thus, any head of D and any head of D' are separated by a large amount.

Refer to page 215 in [3] as to how the constants β and γ have to be chosen.

Observe that two dumbbells that are in the same group may share a head. (This follows, e.g., from Exercise 1.) The next lemma states that, for any two dumbbells from the same group, no head of the longer dumbbell can be properly contained in any head of the shorter dumbbell.

Lemma 1 *Let D and D' be dumbbells that are in the same group, let ℓ and ℓ' be the lengths of D and D', respectively, and assume that $\ell \leq \ell'$. Let $R(A)$ be a head of D and let $R(A')$ be a head of D', and assume that $R(A') \subseteq R(A)$. Then $R(A') = R(A)$.*

Proof. First observe that, by the two properties given above,

$$\ell' \geq \ell/\beta.$$

Assume that $R(A')$ is a proper subset of $R(A)$. Let L be the length of a longest side of $R(A)$, and let s be the separation ratio. It follows from Lemma 11.3.1 in [3] that

$$L \geq \frac{2\ell'}{\sqrt{d}(s+4)}.$$

On the other hand, by Lemma 9.1.2 in [3], we have

$$L \leq 2\ell/s,$$

implying that

$$\ell' \leq \frac{\sqrt{d}(s+4)\ell}{s}.$$

Thus, we have

$$\beta \geq \frac{s}{\sqrt{d}(s+4)},$$

contradicting the choice of β on page 215 in [3]. ∎

3.1 Non-binary dumbbell trees

Consider one of the groups in the partition of the dumbbells. We add the dummy dumbbell to this group. Let D be a non-dummy dumbbell in this group and let ℓ be its length. It follows from the length-grouping and empty-region properties, that there is a unique dumbbell D' of minimal length, say ℓ', in this group such that

- $\ell' > \ell$
- and the distance between some head of D and some head of D' is at most some constant times ℓ. (This constant depends on β, γ, and the separation ratio s.)

Observe that, since $\ell' > \ell$ and D and D' are "close" together, ℓ' is in fact much larger than ℓ.

Informally, if there were another dumbbell D'' of the same length ℓ' that satisfies the second property, then D' and D'' would be "close" together and, therefore, violate the empty-region property. For a formal proof, see Lemma 11.6.1 in [3].

This property implies that each group is indeed "almost" perfectly nested. Consider the dumbbells D and D' as above. Let H and H' be heads of D and D', respectively, that are "close" together. Then we obtain a dumbbell tree by making the dumbbell node storing D a child of the head node storing H', which in turn is made a child of the dumbbell node storing D'. As in Section 2, each point p of S is stored at one leaf; the parent of this leaf is the deepest head node whose head contains p. Note that this is the version of a dumbbell tree as defined in Chapter 11 of [3].

Given the dumbbell tree for each group of dumbbells, we define a graph G as in Section 2. In order to prove that G is a spanner, we need the following result (which was obvious for the dumbbell tree in Section 2):

Lemma 2 (Lemma 11.8.2 in [3]) *Let p and q be two distinct points of S, let $\{A, B\}$ be the pair in the WSPD such that $p \in A$ and $q \in B$, and let D be the dumbbell corresponding to this pair. Consider the dumbbell tree that stores D, and let u be the dumbbell node in this tree that stores D. The left child u_ℓ of u is a head node storing $R(A)$, whereas the right child u_r of u is a head node storing $R(B)$. Let v and w be the leaves storing p and q, respectively. Then v is in the subtree of u_ℓ and w is in the subtree of u_r.*

Using this lemma, the same analysis as in Section 2 shows that G is a spanner. For complete proofs, see Sections 11.6–11.9 in [3].

3.2 Binary dumbbell trees

Consider one of the dumbbell trees DT as defined in Section 3.1. Even though each dumbbell node in DT has two children, a head node may have many dumbbell nodes and leaves as its children. In this section, we will show how to convert DT to a binary dumbbell tree DT' that can, as before, be used to obtain a spanner of the point set S.

Let u be a head node in DT and let H be the head stored at u. This head node has two types of children:

- Dumbbell nodes, storing dumbbells $D_1, D_2, \ldots$ Each such dumbbell D_i has a head that is close to H. (Note that this head of D_i may be contained in H; in fact, it may even be equal to H. By Lemma 1, however, H is not properly contained in this head of D_i.)
- Leaf nodes, storing points $p_1, p_2, \ldots$ Each such point p_i is contained in H.

For each dumbbell D_i, choose a point q_i (of S) in one of its heads. Then do the following:

- Construct a split tree T_H for the points $p_1, p_2, \ldots, q_1, q_2, \ldots$; recall that T_H is a binary tree.
- In DT, replace the node u by the split tree T_H. Each leaf of T_H storing a point q_i is replaced by the dumbbell node storing D_i.

After we have done this for all head nodes u in DT, we obtain the binary dumbbell tree DT'. As before, DT' contains dumbbell nodes, head nodes, and leaves. Additionally, the tree contains *split nodes*, which are nodes that are in some split tree T_H. For each node v of DT', we choose a representative $rep(v)$, which is an arbitrary point of S stored in its subtree.

The graph G is defined as before: For every binary dumbbell tree DT', and for every pair (u, v) of nodes in DT', where u is a child of v, the graph G contains the edge $\{rep(u), rep(v)\}$.

The proof that G is a spanner uses the same argument as before, together with the following fact (recall that d denotes the dimension):

- In any split tree, if we walk down d levels, then the longest side of the bounding box stored at the nodes shrinks by a factor of at least 2. (See Lemma 9.5.3 in [3].) Therefore, the length of a path in G obtained from the representatives of the split nodes on a path within a split tree T_H forms a geometric series. This length is proportional to ℓ/s, where ℓ is the length of the dumbbell that has H as a head and s is the separation ratio of the WSPD. (To prove this, we need that all points in T_H are contained in a box with sidelengths $O(\ell/s)$. This is proved in Lemma 11 in [2].)

Since each binary dumbbell tree DT' contains split trees T_H for different subsets of S, it is not clear how to obtain a good upper bound on the weight of the spanner G. In the next section, we will define a modified binary dumbbell tree for which we can obtain a good weight bound.

4 The final binary dumbbell trees

Consider again one of the groups in the partition of the dumbbells. As before, we add the dummy dumbbell to this group. Recall that T denotes the split tree for the point set S. We will show how to construct a binary dumbbell tree BDT for this group.

We start by sorting the dumbbells in this group by their lengths. Then we "process" the dumbbells one by one, in this sorted order, and maintain a forest of binary trees. During the processing, new trees are added to the forest and trees are merged so that, at the end, the forest consists of one single binary tree, which is the binary dumbbell tree BDT.

Consider all dumbbells that have already been processed. We consider the union of these dumbbells as a geometric region in $\mathbb{R}^d$. This region consists of one or more connected components. During the processing, the following invariant will be maintained:

Invariant: Consider the connected components of all dumbbells that have already been processed. Each such component is stored in a binary tree.

The processing starts with an empty forest. Let D be the current dumbbell to be processed, and let $\{A, B\}$ be the pair in the WSPD that resulted in D. Processing D consists of the following:

- Create a dumbbell node for D.
- Create two head nodes, one for $R(A)$ and one for $R(B)$, and make them the children of the new dumbbell node.
- To obtain the children for the head node for $R(A)$, do the following:
  - Take all trees $T_1, \ldots, T_a$ in the current forest such that each such T_i stores at least one head that overlaps $R(A)$.

    Comment: Let H_i be such a head that is stored in T_i. It follows from properties of the split tree that $H_i \subseteq R(A)$ or $R(A) \subseteq H_i$. By Lemma 1, the latter cannot happen. Thus, we have $H_i \subseteq R(A)$.
  - Let u be the node in the split tree T that stores $R(A)$.
  - For each i with $1 \leq i \leq a$, let v_i be a highest node in the subtree of u such that the bounding box stored at v_i is stored at some head node in T_i.
  - Take all points $p_{a+1}, \ldots, p_b$ in S such that each such p_i is contained in $R(A)$ but p_i is not contained in any head of any dumbbell that has already been processed.
  - For each i with $a < i \leq b$, let v_i be the leaf in the split tree T that stores p_i.

    Comment: Each such v_i is in the subtree of u.
  - **Comment:** The subtrees of the nodes $v_1, \ldots, v_b$ are pairwise disjoint.
  - For each i with $1 \leq i \leq b$, let P_i be the path in T between u and v_i.
  - Let T^u be the tree obtained by taking the union of all paths P_i, $1 \leq i \leq b$.
  - Make the root of the binary tree T^u a child of the head node storing $R(A)$.
  - For each i with $1 \leq i \leq a$, make the root of T_i a child of v_i.
  - **Comment:** It may happen that $v_i = u$ for some i with $1 \leq i \leq a$. If this is the case, then $a = 1$ (and, thus, $i = 1$), $b = 0$, and T^u consists of the single node u.
- In a symmetric way, obtain the children for the head node for $R(B)$.

After all dumbbells have been processed, we have one binary tree for all dumbbells in this group of the partition of all dumbbells. If a node has only one child, then we contract this node-child pair into one single node. The resulting tree is the binary dumbbell tree BDT, in which each internal node has exactly two children.

The graph G corresponding to all dumbbell trees BDT is defined as before. Using the same analysis, it follows that G is a spanner.

We now explain how the binary dumbbell tree BDT can be obtained in $O(n \log n)$ time. The algorithm maintains the following invariant:

Invariant:

- Consider the connected components of all dumbbells that have already been processed. Each such component is stored in a binary tree.
- Consider all dumbbells that have already been processed. For each such dumbbell D', let $\{A', B'\}$ be the pair in the WSPD that defines D'. In the split tree T, the two nodes storing the bounding boxes $R(A')$ and $R(B')$ are *marked*.
- All marked nodes in T are stored in a *union-find* data structure. If such a marked node stores the box $R(C')$, then the "name" of the set containing this node is a pointer to the root of the tree in the current forest storing the head $R(C')$.

At the start of the algorithm, we have an empty forest and none of the nodes in the split tree is marked.

Let D be the current dumbbell to be processed, let $\{A, B\}$ be the corresponding pair in the WSPD, and let u be the node in the split tree T that stores $R(A)$. Consider the subtree T_u of T rooted at u. Define T'_u to be the tree obtained by deleting from T_u all nodes that have a proper ancestor that is marked. (This tree T'_u is obtained by a post-order traversal of T_u, stopping at each leaf and each marked node.) Using the union-find data structure, we obtain the nodes $v_1, \ldots, v_a$ and the trees $T_1, \ldots, T_a$ defined above. Moreover, the leaves in T'_u that have not been marked give us the points $p_{a+1}, \ldots, p_b$ and the corresponding nodes $v_{a+1}, \ldots, v_b$. This information allows us to perform the steps described above. Afterwards, we mark node u and take the union of all sets of marked nodes involved.

During the entire algorithm, the split tree is traversed only once. This, together with the use of the union-find data structure implies that the running time is $O(n \log n)$.

4.1 Bounding the weight of the spanner

In this section, we show how to obtain an upper bound on the weight of the spanner G that is obtained from the binary dumbbell trees BDT.

Let G_{WSPD} be the graph with vertex set S, whose edge set contains, for each pair $\{A, B\}$ in the WSPD, one arbitrary edge having one vertex in A and the other vertex in B.

For each node u of the split tree T, let S_u denote the set of points in S that are stored at the leaves of u's subtree. Recall that each such node u stores the bounding box $R(S_u)$ of

S_u. Also, recall that the size $size(R(S_u))$ of this box is defined to be the length of its longest side. Define

$$W_T = \sum_{u \in T} size(R(S_u)).$$

The weight of the spanner G is proportional to the sum of

- the weight of G_{WSPD} and
- the value W_T.

In the following two subsections, we will show that both these quantities are $O(\log n)$ times the weight of a minimum spanning tree of S.

4.1.1 Bounding the weight of G_{WSPD}

There is a one-to-one correspondence between the edges of G_{WSPD} and the dumbbells. Moreover, the length of each such edge is proportional to the length of the corresponding dumbbell.

Consider again the partition of the dumbbells into $O(1)$ groups; see Section 3. Let E be the subset of the edge set of G_{WSPD} that corresponds to the dumbbells in one group in this partition.

Let D be the length of a longest edge in E. Then all edges in E of length at most D/n have total weight $O(n) \times D/n = O(D)$, which is proportional to the weight of a minimum spanning tree of S.

Partition the edges in E having length more than D/n into $O(\log n)$ *buckets*: One bucket for edges whose lengths are in $(D/2, D]$, one bucket for edges whose lengths are in $(D/4, D/2]$, one bucket for edges whose lengths are in $(D/8, D/4]$, etc.

By the empty-region property, the edges in any bucket satisfy the *gap property*. Therefore, by Lemma 5 in [4], their total weight is proportional to the weight of a minimum spanning tree of S.

Thus, since there are $O(\log n)$ buckets, the total weight of the edge set E is $O(\log n)$ times the weight of a minimum spanning tree of S. Since the number of groups in the partition of the dumbbells is $O(1)$, the same upper bound holds for the weight of the graph G_{WSPD}.

4.1.2 Bounding the value of W_T

The following two properties hold for the split tree T:

- If we walk down d levels in T, then the longest side of the box stored at the nodes shrinks by a factor of at least 2. (See Lemma 9.5.3 in [3].)
- Let u be a node in T, and let ℓ be a real number such that the size of $R(S_u)$ is in the interval $[\ell, 2\ell]$. Consider the set N_u of all nodes v in T such that the size of $R(S_v)$ is in $[\ell, 2\ell]$ and the distance between $R(S_u)$ and $R(S_v)$ is at most $c\ell$, where c is some constant. Then the set N_u has size $O(1)$.

Proof. Let v be a node in N_u. By the first property, and since the split tree is binary, there are $2^{O(d)} = O(1)$ nodes in the subtree of v that belong to N_u. Thus, it suffices to estimate the number of nodes in N_u having the property that their subtrees are pairwise disjoint, i.e., their bounding boxes are pairwise non-overlapping. By Lemma 9.4.3 in [3], this number is O(1). ∎

As we did for the dumbbells in Section 3, we can use the second property to partition the bounding boxes $R(S_u)$, where u ranges over all nodes of the split tree, into $O(1)$ groups such that the following two properties hold:

- If two boxes are in the same group, then either they have approximately the same size (i.e., within a factor of 2 of each other), or the size of one box is larger than the size of the other box by at least some large constant multiplicative factor.
- If two distinct boxes B and B' are in the same group, and if their sizes are in the interval $[\ell, 2\ell]$, then the distance between B and B' is at least some large constant times ℓ.

Consider one group in the partition, and let B be a box in this group. Consider the opposite faces of B whose distance is equal to the size of B. The point set S contains two points p_B and q_B on these opposite faces. (Thus, if $d = 2$ and the horizontal side defines the size of B, then p_B and q_B are on the vertical sides.)

Let G' be the graph with vertex set S consisting of all edges $\{p_B, q_B\}$. The weight of G' is proportional to the value

$$\sum_u \mathit{size}(R(S_u)),$$

where the sum is over all nodes u for which $R(S_u)$ is in the group of the partition that we are considering. By the same proof technique as in Section 4.1.1, the weight of G' is $O(\log n)$ times the weight of a minimum spanning tree of S. Since the number of groups in the partition of the boxes is $O(1)$, we obtain the same upper bound for the summation

$$W_T = \sum_{u \in T} \mathit{size}(R(S_u)).$$

4.2 Obtaining a bounded-degree spanner

The spanner G defined above depends on the way we choose the representative point $\mathit{rep}(u)$, for any node u in any binary dumbbell tree BDT. For any choice of representatives, we obtain a spanner whose weight is $O(\log n)$ times the weight of a minimum spanning tree of S.

Consider one of the binary dumbbell-trees BDT. We now choose the representatives in the following way:

- The representative $\mathit{rep}(u)$ of a leaf u is the point stored at the leaf.

- Let u be an internal node. The representative $rep(u)$ of u is the point stored at the rightmost leaf in the left subtree of u.

For each point p in S, there are at most two nodes u in BDT such that $rep(u) = p$; one of these nodes is a leaf. It follows that, in the subgraph of the spanner G that is implied by BDT, p has degree at most four. Thus, since the number of dumbbell trees is $O(1)$, each node in the complete spanner G has degree $O(1)$.

5 Conclusion

Given a set S of n points in $\mathbb{R}^d$ and given a real constant $\epsilon > 0$, we can compute, in $O(n \log n)$ time, a collection of $O(1)$ binary dumbbell trees. For any two distinct points p and q in S, there is a binary dumbbell tree BDT for which the following holds: Let u and v be the leaves of BDT that store p and q. Then the path joining the representatives of the nodes on the path in BDT between u and v has length at most $(1+\epsilon)|pq|$. The collection of dumbbell trees defines a spanner, whose weight is $O(\log n)$ times the weight of a minimum spanning tree of S. Moreover, the maximum degree of any point in the spanner is $O(1)$.

References

[1] S. Arya, G. Das, D. M. Mount, J. S. Salowe, and M. Smid. Euclidean spanners: short, thin, and lanky. In *Proceedings of the 27th ACM Symposium on the Theory of Computing*, pages 489–498, 1995.

[2] P. Carmi and M. Smid. An optimal algorithm for computing angle-constrained spanners. In *Proceedings of the 21st International Symposium on Algorithms and Computation, Part I*, volume 6506 of *Lecture Notes in Computer Science*, pages 316–327, Berlin, 2010. Springer-Verlag.

[3] G. Narasimhan and M. Smid. *Geometric Spanner Networks*. Cambridge University Press, Cambridge, UK, 2007.

[4] M. Smid. The weak gap property in metric spaces of bounded doubling dimension. In *Efficient Algorithms, Essays Dedicated to Kurt Mehlhorn on the Occasion of His 60th Birthday*, volume 5760 of *Lecture Notes in Computer Science*, pages 275–289, Berlin, 2009. Springer-Verlag.