The Well-Separated Pair Decomposition and Its Applications

Michiel Smid

School of Computer Science

Carleton University

Ottawa, Ontario, Canada, K1S 5B6

E-mail: michiel@scs.carleton.ca

December 22, 2016

1.1 Introduction

Computational geometry is concerned with the design and analysis of algorithms that solve problems on geometric data in $\mathbb{R}^d$, where the dimension d is a constant. A large part of the field has been devoted to problems that involve distances determined by pairs of points in a given point set. Given a set S of n points in $\mathbb{R}^d$, we may wish to compute a pair p, q of distinct points in S whose distance is minimum, the k smallest distances among the $\binom{n}{2}$ pairwise distances, the nearest neighbor of each point of S, or the minimum spanning tree of S. Most problems of this type can be rephrased as a graph problem on the complete Euclidean graph on S, in which each edge pq has a weight being the Euclidean distance $|pq|$ between p and q. Since the number of edges in this graph is $\Theta(n^2)$, many problems involving pairwise distances can trivially be solved in $O(n^2)$ time. Even though the complete Euclidean graph has size $\Theta(n^2)$, it can be represented in $\Theta(n)$ space: It is clearly sufficient to only store the points of S, because the weight of any edge can be computed in $O(1)$ time. This leads to the question whether distance problems can be solved in subquadratic time, possibly at the cost of obtaining an approximate solution. For many of these problems, subquadratic algorithms have indeed been designed; see for example the books by Preparata and Shamos [21] and de Berg *et al.* [11], and the survey papers by Bern and Eppstein [5], Eppstein [12], and

Smid [25]. Most of these algorithms, however, are tailored to the problem at hand.

Callahan and Kosaraju [8, 10] devised the *well-separated pair decomposition (WSPD)*, and showed that it can be used to solve a large variety of distance problems. Intuitively, a WSPD is a partition of the $\binom{n}{2}$ edges of the complete Euclidean graph into $O(n)$ subsets. Each subset in this partition is represented by two subsets A and B of the point set S, such that (i) all distances between points in A and points in B are approximately equal, (ii) all distances within the point set A are much smaller than distances between A and B, and (iii) all distances within the point set B are much smaller than distances between A and B. Thus, a WSPD can be regarded as a set of $O(n)$ edges that approximates the dense complete Euclidean graph.

Callahan and Kosaraju showed that the WSPD can be used to obtain optimal algorithms for solving the closest pair problem, the k closest pairs problem, the all-nearest neighbors problem, and the approximate minimum spanning tree problem. After the publication of this influential paper, other researchers have shown that the WSPD can be used to solve many other problems. In this paper, we give an overview of several proximity problems that can be solved efficiently using the WSPD. We mention that the WSPD has also been a critical tool for the solution of several variants of the problem of constructing spanners; an overview of these results can be found in the chapter by Gudmundsson and Knauer [16] in this handbook. An extensive treatment of the WSPD and its applications is given in the book by Narasimhan and Smid [20].

The rest of this paper is organized as follows. In Section 1.2, we define the WSPD. In Section 1.3, we present an efficient algorithm for constructing a WSPD. In Section 1.4, we show that the WSPD can be used to obtain optimal algorithms for the closest pair problem, the k closest pairs problem, and the all-nearest neighbors problem. In Section 1.5, we use the WSPD to obtain approximate solutions for the diameter problem, the spanner problem, the minimum spanning tree problem, and the problem of computing the k-th closest pair. Finally, in Section 1.6, we mention some results on generalizing the WSPD to more general metric spaces.

1.2 Well-Separated Pairs

In this section, we define the well-separated pair decomposition (WSPD), and prove one of its main properties in Lemma 1.1. As mentioned in Section 1.1, the WSPD was introduced by Callahan and Kosaraju [8, 10]. Previously, however, similar ideas were used by Salowe [23, 24] and Vaidya [28, 29, 30], who designed efficient algorithms for computing spanners, all-nearest neighbors, and k closest pairs.

We start by defining the notion of two sets being well-separated. For any set X of points in $\mathbb{R}^d$, we denote its *bounding box* by $R(X)$. Thus, $R(X)$ is the smallest axes-parallel hyperrectangle that contains the set X.

Definition 1.1 *Let A and B be two finite sets of points in $\mathbb{R}^d$ and let $s > 0$ be a real number. We say that A and B are* well-separated *with respect to s, if there exist two disjoint balls C_A and C_B, such that*

1. *C_A and C_B have the same radius,*
2. *C_A contains $R(A)$,*
3. *C_B contains $R(B)$, and*
4. *the distance between C_A and C_B is at least s times the radius of C_A.*

The real number s is called the separation ratio.

If we are given the bounding boxes $R(A)$ and $R(B)$ of the sets A and B, respectively, then we can test in $O(1)$ time whether these two sets are well-separated.

In the next lemma, we prove the two properties of well-separated sets that were mentioned already in Section 1.1: If A and B are well-separated with respect to a large separation ratio s, then (i) all distances between points in A and points in B are approximately equal and (ii) all distances between points in A (resp. B) are much smaller than distances between A and B. These two properties will be used repeatedly in the rest of this paper.

Lemma 1.1 *Let $s > 0$ be a real number, let A and B be two sets in $\mathbb{R}^d$ that are well-separated with respect to s, let a and a' be two points in A, and let b and b' be two points in B. Then, we have*

1. $|aa'| \leq (2/s)|ab|$, *and*

2. $|a'b'| \leq (1 + 4/s)|ab|$.

Proof: Let C_A and C_B be disjoint balls of the same radius, say ρ, such that $R(A) \subseteq C_A$, $R(B) \subseteq C_B$, and the distance between C_A and C_B is at least $s\rho$. The first claim follows from the facts that $|aa'| \leq 2\rho$ and $|ab| \geq s\rho$. By combining the first claim with the triangle inequality, we obtain

$$|a'b'| \leq |a'a| + |ab| + |bb'| \leq (1 + 4/s)|ab|,$$

proving the second claim. □

Definition 1.2 *Let S be a set of n points in $\mathbb{R}^d$, and let $s > 0$ be a real number. A* well-separated pair decomposition (WSPD) *for S, with respect to s, is a sequence*

$$\{A_1, B_1\}, \{A_2, B_2\}, \ldots, \{A_m, B_m\}$$

of pairs of non-empty subsets of S, for some integer m, such that

1. *for each i with $1 \leq i \leq m$, A_i and B_i are well-separated with respect to s, and*

2. *for any two distinct points p and q of S, there is exactly one index i with $1 \leq i \leq m$, such that*

   (a) *$p \in A_i$ and $q \in B_i$, or*

   (b) *$p \in B_i$ and $q \in A_i$.*

The integer m is called the size *of the WSPD.*

Observe that a WSPD always exists: If we let any two distinct points p and q of S form a pair $\{\{p\}, \{q\}\}$, then the conditions in Definition 1.2 are satisfied. The size of this WSPD, however, is $\binom{n}{2}$. In the next section, we will give an algorithm that computes a WSPD whose size is only $O(n)$.

We remark that, for any set of n points in $\mathbb{R}^d$, the size m of any WSPD satisfies $m \geq n - 1$. An elegant proof of this fact, using linear algebra, was given by Graham and Pollak [15]; see also Chapter 9 in Aigner and Ziegler [2]. Moreover, for any set of n points in $\mathbb{R}^d$ and for any WSPD, the total size $\sum_i (|A_i| + |B_i|)$ of all sets in the decomposition is $\Omega(n \log n)$; see Bollobás and Scott [7]. Callahan and Kosaraju have shown that, for some point sets, this summation is, in fact, $\Omega(n^2)$ for any WSPD.

1.3 Computing a well-separated pair decomposition

Let S be a set of n points in $\mathbb{R}^d$, and let $s > 0$ denote the separation ratio. The algorithm that constructs a WSPD for S consists of two phases. In the first phase, a so-called split tree is constructed, which can be considered to be a hierarchical decomposition of the bounding box of S into axes-parallel hyperrectangles. In the second phase, the split tree is used to actually compute the WSPD. The algorithm is due to Callahan and Kosaraju [10].

1.3.1 The split tree

The *split tree* $T(S)$ for the point set S is a binary tree that is defined as follows:

1. If $n = 1$, then $T(S)$ consists of a single node storing the only element of S.
2. Assume that $n \geq 2$ and consider the bounding box $R(S) = \prod_{i=1}^{d} [\ell_i, r_i]$ of S. Let i be the dimension such that $r_i - \ell_i$ is maximum, and define $S_1 := \{p \in S : p_i \leq (\ell_i + r_i)/2\}$ and $S_2 := S \setminus S_1$. The split tree $T(S)$ for S consists of a root whose two children are recursively defined split trees $T(S_1)$ and $T(S_2)$ for the sets S_1 and S_2, respectively. The root of $T(S)$ stores the bounding box $R(S)$.

Thus, the split tree $T(S)$ stores the points of S at its leaves. Each internal node of $T(S)$ stores an axes-parallel hyperrectangle, which is the bounding box of the set of all points of S that are stored in its subtree. Observe that the split tree is, in general, not balanced.

The above definition immediately leads to an $O(n^2)$–time algorithm for constructing the split tree. Callahan and Kosaraju show that, by using a divide-and-conquer approach, the split tree can in fact be constructed in $O(n \log n)$ time:

Lemma 1.2 *The split tree for any set of n points in $\mathbb{R}^d$ can be constructed in $O(n \log n)$ time.*

1.3.2 Using the split tree to compute well-separated pairs

Consider the split tree $T = T(S)$ for the point set S. For any node u of T, we denote by S_u the subset of S that is stored at the leaves of the subtree rooted at u. For any subset X of $\mathbb{R}^d$, we denote by $L_{\max}(R(X))$ the length of a longest side of the bounding box $R(X)$ of X.

The following algorithm uses the split tree to compute a WSPD for S. Recall that s denotes the separation ratio.

Step 1: Initialize an empty queue Q. For each internal node u of T, do the following: Let v and w be the two children of u. Insert the pair (v, w) into Q.

Step 2: Repeat the following until the queue Q is empty: Take the first pair (v, w) in Q and delete it from Q. If the sets S_v and S_w are well-separated with respect to s, then output the pair $\{S_v, S_w\}$. Otherwise, assume without loss of generality that $L_{\max}(R(S_v)) \leq L_{\max}(R(S_w))$. Let w_1 and w_2 be the two children of w. Insert the pairs (v, w_1) and (v, w_2) into the queue Q.

It is not difficult to see that the output of this algorithm is a well-separated pair decomposition of the point set S. Callahan and Kosaraju use a non-trivial packing argument to show that the number of pairs is $O(n)$.

Lemma 1.3 *Given the split tree, the above algorithm constructs, in $O(s^d n)$ time, a WSPD for S that consists of $O(s^d n)$ pairs.*

Observe that the WSPD is represented implicitly by the split tree: Each pair $\{A, B\}$ in the WSPD is represented by two nodes v and w, such that $A = S_v$ and $B = S_w$. Thus, the entire WSPD can be represented using $O(s^d n)$ space.

By combining Lemmas 1.2 and 1.3, we obtain the following result:

Theorem 1.1 (Callahan and Kosaraju [10]) *Given a set S of n points in $\mathbb{R}^d$, and given a real number $s > 0$, a well-separated pair decomposition for S, with separation ratio s, consisting of $O(s^d n)$ pairs, can be computed in $O(n \log n + s^d n)$ time.*

In some applications, it is useful to have a WSPD in which each well-separated pair consists of two sets, at least one of which is a singleton set. For the WSPD $\{A_i, B_i\}$, $1 \leq i \leq m$, that is constructed by the algorithm given above, Callahan [8] proves that

$$\sum_{i=1}^{m} \min(|A_i|, |B_i|) = O\left(s^d n \log n\right).$$

Thus, if we replace each pair $\{A_i, B_i\}$ (where we assume without loss of generality that $|A_i| \leq |B_i|$) by $|A_i|$ pairs $\{\{a\}, B_i\}$, $a \in A_i$, then we obtain the following result:

Theorem 1.2 (Callahan [8]) *Let S be a set of n points in $\mathbb{R}^d$, and let $s > 0$ be a real number. In $O(s^d n \log n)$ time, a well-separated pair decomposition for S, with separation ratio s, can be constructed, such that each well-separated pair consists of two sets, at least one of which is a singleton set, and the total number of pairs is $O(s^d n \log n)$.*

1.4 Exact algorithms for proximity problems

As we have mentioned before, the WSPD is an $O(n)$–size approximation of the set of $\Theta(n^2)$ distances determined by a set of n points. In this section, we show that, despite the fact that the WSPD *approximates* all distances, it can be used to solve several proximity problems *exactly*. All results in this section are due to Callahan and Kosaraju [8, 10].

Let S be a set of n points in $\mathbb{R}^d$, and let $s > 0$ be a real number. Consider the split tree T and the corresponding WSPD for S, with separation ratio s, consisting of the pairs $\{A_i, B_i\}$, $1 \leq i \leq m$.

1.4.1 The closest pair problem

In this problem, we want to compute a *closest pair* in S, i.e., two distinct points p and q in S such that $|pq|$ is minimum. Many algorithms are known that solve this problem optimally in $O(n \log n)$ time; see Smid [25]. We show that the WSPD "contains" a solution to the closest pair problem.

Let (p, q) be a closest pair in S, and let i be the index such that $p \in A_i$ and $q \in B_i$. If we assume that the separation ratio s is a constant larger than two, then it follows from Lemma 1.1 that both A_i and B_i are singleton sets. Thus, by considering all pairs $\{A_j, B_j\}$, $1 \leq j \leq m$, such that both A_j and B_j are represented by leaves of the split tree, we obtain the closest pair in $O(m)$ time. Combining this with Theorem 1.1, we obtain the following result:

Theorem 1.3 *Given a set S of n points in $\mathbb{R}^d$, a closest pair in S can be computed in $O(n \log n)$ time.*

1.4.2 The k closest pairs problem

We next consider the problem of computing the k *closest pairs* in S, for any given integer k with $1 \leq k \leq \binom{n}{2}$. That is, we want to compute the k smallest elements in the (multi)set of $\binom{n}{2}$ distances determined by pairs of points in S. Several algorithms have been designed that solve this problem optimally in $O(n \log n + k)$ time; see Smid [25]. As for the closest pair problem, we show that, once the WSPD is given, the k closest pairs can be obtained in a simple way.

For each i with $1 \leq i \leq m$, we denote by $|R(A_i)R(B_i)|$ the minimum distance between the bounding boxes $R(A_i)$ and $R(B_i)$ of A_i and B_i, respectively. We assume, for ease of notation, that the pairs in the WSPD are numbered such that

$$|R(A_1)R(B_1)| \leq |R(A_2)R(B_2)| \leq \ldots \leq |R(A_m)R(B_m)|.$$

(This ordering of the pairs is only used in the analysis, it is *not* computed by the algorithm.) The algorithm does the following:

Step 1: Compute the smallest integer $\ell \geq 1$, such that $\sum_{i=1}^{\ell} |A_i| \cdot |B_i| \geq k$.

Step 2: Compute the distance r between the bounding boxes $R(A_\ell)$ and $R(B_\ell)$ of the sets A_ℓ and B_ℓ, respectively.

Step 3: Compute the largest index ℓ' such that $|R(A_{\ell'})R(B_{\ell'})| \leq (1 + 4/s)r$.

Step 4: Compute the set L' consisting of all pairs (p, q) for which there is an index i with $1 \leq i \leq \ell'$, such that $p \in A_i$ and $q \in B_i$.

Step 5: Return the k smallest distances determined by the pairs in the set L'.

Lemma 1.4 *This algorithm computes the k closest pairs in S.*

Proof: Let (p, q) be one of the k closest pairs, and let j be the index such that $p \in A_j$ and $q \in B_j$. It suffices to prove that (p, q) is an element of L', i.e., $j \leq \ell'$. To prove this, assume that $j > \ell'$. Then

$$|pq| \geq |R(A_j)R(B_j)| > (1 + 4/s)r.$$

Let L be the set consisting of all pairs (x, y) for which there is an index i with $1 \leq i \leq \ell$, such that $x \in A_i$ and $y \in B_i$. This set contains at least k elements. Using Lemma 1.1, we have, for each pair (x, y)

in L,

$$|xy| \leq (1+4/s)|R(A_i)R(B_i)| \leq (1+4/s)|R(A_\ell)R(B_\ell)| = (1+4/s)r.$$

This contradicts our assumption that (p,q) is one of the k closest pairs. □

Using a linear-time (weighted) selection algorithm, the running time of the algorithm can be bounded by

$$O\left(m + \sum_{i=1}^{\ell'} |A_i| \cdot |B_i|\right) = O(m + |L'|).$$

Let δ be the k-th smallest distance in S. Then it can be shown that $r \leq \delta$ and $|pq| \leq (1+4/s)^2\delta$, for any pair (p,q) in L'. Hence, if we denote by M the number of distances in the set S that are at most $(1+4/s)^2\delta$, then the running time of the algorithm is $O(m+M)$. By using a counting technique, based on a grid with cells having sides of length $\delta/\sqrt{d}$, it can be shown that

$$M = O\left((1+4/s)^{2d}(n+k)\right).$$

We take the separation ratio s to be equal to, say, one. Then, by combining our results with Theorem 1.1, we obtain the following theorem:

Theorem 1.4 *Given a set S of n points in $\mathbb{R}^d$, and given an integer k with $1 \leq k \leq \binom{n}{2}$, the k closest pairs in S can be computed in $O(n \log n + k)$ time.*

1.4.3 The all-nearest neighbors problem

In this problem, we want to compute for each point p of S a *nearest neighbor* in S, i.e., a point $q \in S \setminus \{p\}$ for which $|pq|$ is minimum. Vaidya [29] was the first to solve this problem optimally in $O(n \log n)$ time. In fact, his algorithm uses ideas that are very similar to the WSPD. In this section, we sketch the algorithm of Callahan and Kosaraju [10].

Let p be a point of S and let q be its nearest neighbor. Let i be the index such that $p \in A_i$ and $q \in B_i$. It follows from Lemma 1.1 that the set A_i consists only of the point p. Hence, in order to solve the all-nearest neighbors problem, we only have to consider pairs of the WSPD, for which at least one of their sets is a singleton set. This observation does not lead to an efficient algorithm yet.

For any node u of the split tree T, we define $F(u)$ to be the set of all points $p \in S$ such that $\{\{p\}, S_v\}$ is a pair in the WSPD, for some ancestor v of u. (We consider u to be an ancestor of itself.) Also, we define $N(u)$ to be the set of all points $p \in F(u)$, such that the distance from p to the smallest ball containing $R(S_u)$ is at most equal to the smallest distance between p and any other point of $F(u)$. Observe that $N(u) \subseteq F(u)$.

Lemma 1.5 *The size of the set $N(u)$ is $O((s/(s-1))^d)$.*

Proof: Let C be the smallest ball that contains $R(S_u)$. We claim that

1. for each $p \in N(u)$, the sets $\{p\}$ and S_u are well-separated, and
2. $|pC| \leq |pq|$, for any two distinct points p and q of $N(u)$.

By combining these two claims with a generalization of the fact that any point can be the nearest neighbor of at most a constant number of other points, it can be shown that the size of $N(u)$ is $O((s/(s-1))^d)$.

To prove the first claim, let $p \in N(u)$. Since $p \in F(u)$, there is an ancestor v of u such that the sets $\{p\}$ and S_v are well-separated. Since S_u is a subset of S_v, the sets $\{p\}$ and S_u are well-separated as well.

To prove the second claim, let p and q be two distinct points of $N(u)$. The definition of $N(u)$ implies that the distance between p and C is at most the smallest distance between p and any other point of $F(u)$. In particular, since $q \in F(u)$, we have $|pC| \leq |pq|$. □

We assume from now on that the separation ratio s is a constant larger than two. The sets $N(u)$, where u ranges over all nodes of the split tree T, can be computed in a top-down fashion, in $O(n)$ total time. How do we use these sets? Let p be any point of S, let q be a nearest neighbor of p, and let u be the leaf of the split tree that stores q. Let i be the index such that $A_i = \{p\}$ and $q \in B_i$, and let v be the ancestor of u such that $B_i = S_v$. Then, $p \in F(u)$. Moreover, since $S_u = \{q\}$, the distance between p and the smallest ball containing $R(S_u)$ is equal to $|pq|$, which is at most the distance between p and any other point of $F(u)$. Therefore, we have $p \in N(u)$.

The discussion above, together with Theorem 1.1, leads to an algorithm that solves the all-nearest neighbors problem in $O(n \log n)$ time.

Theorem 1.5 *Given a set S of n points in $\mathbb{R}^d$, the all-nearest neighbors problem can be solved in $O(n \log n)$ time.*

1.5 Approximation algorithms for proximity problems

In this section, we consider proximity problems for which no optimal exact algorithms are known. For each of these problems, we show that the WSPD leads to simple and fast approximation algorithms.

Let S be a set of n points in $\mathbb{R}^d$, and let $s > 0$ be a real number. We assume that we have already computed the split tree T and the corresponding WSPD for S, with separation ratio s, consisting of the pairs $\{A_i, B_i\}$, $1 \leq i \leq m$. For each i with $1 \leq i \leq m$, we choose an arbitrary element a_i in A_i, and an arbitrary element b_i in B_i.

1.5.1 The diameter problem

The *diameter* of S is defined to be the largest distance between any two points of S. If the dimension d is equal to two, the diameter can be computed in $O(n \log n)$ time; see Preparata and Shamos [21]. Ramos [22] obtained the same time bound for the three-dimensional case. It is not known if for dimensions larger than three, the diameter can be computed in $O(n \log^{O(1)} n)$ time. In this section, we show that the WSPD leads to a simple and efficient algorithm that approximates the diameter, for any constant dimension d.

Let D be the diameter of S, and let i be the index for which $|a_i b_i|$ is maximum. A straightforward application of Lemma 1.1 shows that $D/(1 + 4/s) \leq |a_i b_i| \leq D$. Hence, if we choose $s = 4(1 - \epsilon)/\epsilon$, then this result, together with Theorem 1.1, yields the following theorem.

Theorem 1.6 *Given a set S of n points in $\mathbb{R}^d$, and given a real constant $0 < \epsilon < 1$, a $(1-\epsilon)$-approximation to the diameter of S can be computed in $O(n \log n)$ time.*

We remark that the same approximation factor can be achieved in only $O(n)$ time: Choose $O(1/\epsilon^{d-1}) = O(1)$ vectors such that, for any pair p, q of distinct points in $\mathbb{R}^d$, one of these vectors makes an angle of at most ϵ with the vector $\overrightarrow{pq}$; see Chapter 5 in Narasimhan and Smid [20]. Then, for each of these vectors $\vec{v}$, compute an extreme point a of S in the positive direction $\vec{v}$, compute an extreme point b of S in the negative

direction $-\vec{v}$, and compute the distance $|ab|$. The largest distance $|ab|$ obtained is a $(1-\epsilon)$-approximation to the diameter of S. The running time of this algorithm is $O(n)$. For details, see Janardan [18].

1.5.2 The spanner problem

For a real number $t > 1$, a graph $G = (S, E)$ is called a *t-spanner* for the point set S, if for any two points p and q of S, we have $|pq|_G \leq t|pq|$, where $|pq|_G$ denotes the length of a shortest path in G between p and q. Many algorithms are known that compute spanners; see the chapter by Gudmundsson and Knauer [16] in this handbook, and Narasimhan and Smid [20]. In this section, we show that the WSPD immediately gives a spanner for S consisting of $O(n)$ edges. The construction is due to Callahan and Kosaraju [9].

Lemma 1.6 *Assume that the separation ratio s is larger than four. Define $G = (S, E)$ to be the graph with edge set $E = \{a_i b_i : 1 \leq i \leq m\}$. Then, G is a t-spanner for S, where $t = (s+4)/(s-4)$.*

Proof: The proof is by induction. Consider two points p and q in S. If $p = q$, then obviously $|pq|_G \leq t|pq|$. Assume that $p \neq q$. Moreover, assume that $|xy|_G \leq t|xy|$, for all points x and y in S for which $|xy| < |pq|$. Let i be the index such that $p \in A_i$ and $q \in B_i$. Using Lemma 1.1, we obtain

$$\begin{aligned}
|pq|_G &\leq |pa_i|_G + |a_i b_i|_G + |b_i q|_G \\
&= |pa_i|_G + |a_i b_i| + |b_i q|_G \\
&\leq t|pa_i| + |a_i b_i| + t|b_i q| \\
&\leq (2t/s + (1 + 4/s) + 2t/s)|pq| \\
&= t|pq|.
\end{aligned}$$

□

Observe that, if the separation ratio s goes to infinity, the value $t = (s+4)/(s-4)$ converges to 1. Thus, Theorem 1.1 implies the following result.

Theorem 1.7 *Given a set S of n points in $\mathbb{R}^d$, and given a real constant $t > 1$, a t-spanner for S, consisting of $O(n)$ edges, can be computed in $O(n \log n)$ time.*

1.5.3 The greedy spanner

Let S be a set of n points in $\mathbb{R}^d$ and let $t > 1$ be a real constant. We have seen in Theorem 1.7 that a t-spanner with $O(n)$ edges can be computed in $O(n \log n)$ time. In one of the earliest papers on geometric spanners, Althöfer *et al.* [3] introduced the following simple greedy algorithm for computing a t-spanner:

Algorithm GREEDYSPANNER(S, t)

sort the $\binom{n}{2}$ pairs of distinct points in non-decreasing order of their
distances (breaking ties arbitrarily), and store them in list L;
$E := \emptyset$;
$G := (S, E)$;
for each pair pq in L $(*$ consider pairs in sorted order $*)$
do if $|pq|_G > t|pq|$
  then $E := E \cup \{pq\}$;
    $G := (S, E)$
  endif
endfor;
return the graph G

It is obvious that the output of this algorithm is a t-spanner for the input set S. The following theorem uses the WSPD of S to prove that the number of edges in the greedy spanner is $O(n)$.

Theorem 1.8 *Let S be a set of n points in $\mathbb{R}^d$, let $t > 1$ be a real constant, and assume that the separation ratio s in the WSPD for S satisfies $t = (s+4)/(s-4)$. Then the number of edges in the greedy spanner for S is $O(n)$.*

Proof: The proof will follow from the claim that, for each pair $\{A_i, B_i\}$ in the WSPD for S, the greedy spanner contains at most one edge pq with $p \in A_i$ and $q \in B_i$.

We prove this claim by contradiction. Assume the greedy spanner contains two distinct edges pq and $p'q'$, with $p, p' \in A_i$ and $q, q' \in B_i$. We may assume without loss of generality that pq is before $p'q'$ in the

list L. Consider the moment when the algorithm examines the pair $p'q'$. At this moment, the edge pq is already in the edge set E. Moreover, by Lemma 1.1, $|pp'| \leq (2/s)|p'q'| < |p'q'|$ and, thus, at this moment, we have $|pp'|_G \leq t|pp'|$. By the same argument, we have $|qq'|_G \leq t|qq'|$. It follows that, at this moment,

$$\begin{aligned} |p'q'|_G &\leq |p'p|_G + |pq| + |qq'|_G \\ &\leq t|p'p| + |pq| + t|qq'| \\ &\leq (2t/s + (1 + 4/s) + 2t/s)|p'q'| \\ &= t|p'q'|. \end{aligned}$$

As a result, the algorithm does not add $p'q'$ as an edge to the greedy spanner. This is a contradiction. □

Using geometric arguments, it can be shown that any two edges pq and pq' of the greedy spanner make a "large" angle. This implies that the maximum degree of the greedy spanner is $O(1)$. Also, using a lengthy analysis, it can be shown that the total edge length of the greedy spanner is within a constant factor of the total length of a minimum spanning tree. Proofs of these claims can be found in Narasimhan and Smid [20]. See also Smid [26].

1.5.4 The minimum spanning tree problem

In the two-dimensional case, the minimum spanning tree of S can be computed in $O(n \log n)$ time, by using the fact that it is contained in the Delaunay triangulation of S; see Preparata and Shamos [21] and de Berg *et al.* [11]. For the three-dimensional case, the best known algorithm has a running time that is close to $O(n^{4/3})$; see Agarwal *et al.* [1]. Erickson [13] argues that it is unlikely that this running time can be improved considerably. In this section, we show that, in any constant dimension d, an approximation to the minimum spanning tree can be computed in $O(n \log n)$ time. The algorithm is due to Callahan and Kosaraju [9].

Let $t > 1$ be a real constant and consider an arbitrary t-spanner for S having $O(n)$ edges. Observe that G is a connected graph. Let T be a minimum spanning tree of G.

Lemma 1.7 *T is a t-approximate minimum spanning tree of S.*

Proof: Let T^* be a minimum spanning tree of S, and denote its total edge length by $|T^*|$. Number the

edges of T^* as $e_1, e_2, \ldots, e_{n-1}$. For each i with $1 \leq i \leq n-1$, let P_i be a t-spanner path (in G) between the endpoints of e_i, and denote the length of this path by $|P_i|$. Then,

$$\sum_{i=1}^{n-1} |P_i| \leq \sum_{i=1}^{n-1} t|e_i| = t|T^*|.$$

Let G' be the subgraph of G consisting of the union of the edges of all paths P_i, $1 \leq i \leq n-1$. Then G' is a connected graph on the points of S, and its weight is at most $t|T^*|$. Since the weight of T is at most that of G', the weight of T is at most t times the weight of T^*. □

Since the graph G contains $O(n)$ edges, its minimum spanning tree T can be computed in $O(n \log n)$ time. By combining this with Theorem 1.7, we obtain the following result.

Theorem 1.9 *Given a set S of n points in $\mathbb{R}^d$, and given a real constant $\epsilon > 0$, a $(1+\epsilon)$-approximation to the minimum spanning tree of S can be computed in $O(n \log n)$ time.*

1.5.5 The k-th closest pair problem

In this problem, we are given an integer k with $1 \leq k \leq \binom{n}{2}$, and want to compute the k-th smallest element in the (multi)set of $\binom{n}{2}$ distances determined by pairs of points in S. In the two-dimensional case, the best known algorithm for this problem has a running time that is close to $O(n^{4/3})$; see Katz and Sharir [19]. Again, Erickson [13] argues that it is unlikely that a significantly faster algorithm exists. We show that, for any constant dimension, there is a simple and efficient algorithm that approximates the k-th closest pair. The results in this section are due to Bespamyatnikh and Segal [6].

Let k be any integer with $1 \leq k \leq \binom{n}{2}$. As in Section 1.4.2, we assume that the pairs in the WSPD are numbered such that

$$|R(A_1)R(B_1)| \leq |R(A_2)R(B_2)| \leq \ldots \leq |R(A_m)R(B_m)|.$$

Let $\ell \geq 1$ be the smallest integer such that $\sum_{i=1}^{\ell} |A_i| \cdot |B_i| \geq k$, let x be an arbitrary element of A_ℓ, and let y be an arbitrary element of B_ℓ.

Lemma 1.8 *If δ is the k-th smallest distance in the set S, then $\delta/(1+4/s) \leq |xy| \leq (1+4/s)\delta$.*

Proof: As mentioned in Section 1.4.2, it can be shown that $|R(A_\ell)R(B_\ell)| \leq \delta$. If we combine this fact with

Lemma 1.1, then we obtain

$$|xy| \leq (1 + 4/s)|R(A_\ell)R(B_\ell)| \leq (1 + 4/s)\delta.$$

Let L be the set consisting of all pairs (a, b) for which there is an index i with $1 \leq i \leq \ell$, such that $a \in A_i$ and $b \in B_i$. Let (a, b) be the pair in L for which $|ab|$ is maximum, and let i be the index such that $a \in A_i$ and $b \in B_i$. Observe that $i \leq \ell$. Since L has size at least k, we have $\delta \leq |ab|$. Therefore (again using Lemma 1.1),

$$\delta \leq |ab| \leq (1 + 4/s)|R(A_i)R(B_i)| \leq (1 + 4/s)|R(A_\ell)R(B_\ell)| \leq (1 + 4/s)|xy|.$$

□

Thus, using Theorem 1.1, we obtain the following result.

Theorem 1.10 *Let S be a set of n points in $\mathbb{R}^d$, let k be an integer with $1 \leq k \leq \binom{n}{2}$, let $\epsilon > 0$ be a constant, and let δ be the k-th smallest distance in S. In $O(n \log n)$ time, a pair (x, y) of points in S can be computed for which $(1 - \epsilon)\delta \leq |xy| \leq (1 + \epsilon)\delta$.*

Surprisingly, computing a pair (x, y) of points in S such that $\delta \leq |xy| \leq (1 + \epsilon)\delta$ is more difficult. In order to compute such a pair, we use the WSPD of Theorem 1.2. Thus, the WSPD consists of pairs $\{A_i, B_i\}$, $1 \leq i \leq m$, where each set A_i is a singleton set, say $A_i = \{a_i\}$, and $m = O(n \log n)$.

For each i with $1 \leq i \leq m$, we define $d_i = \min\{|a_i b| : b \in B_i\}$ and $D_i = \max\{|a_i b| : b \in B_i\}$. Assume that the pairs in the WSPD are numbered such that $d_1 \leq d_2 \leq \ldots \leq d_m$. Let $\ell \geq 1$ be the smallest integer such that $\sum_{i=1}^{\ell} |B_i| \geq k$, and let $D = \max(D_1, D_2, \ldots, D_\ell)$.

Lemma 1.9 *If δ is the k-th smallest distance in the set S, then $\delta \leq D \leq (1 + 4/s)\delta$.*

Proof: Since $\sum_{i=1}^{\ell} |B_i| \geq k$, the pairs $\{A_i, B_i\}$ with $1 \leq i \leq \ell$, define at least k distances. Since D is the largest among these distances, it follows that $\delta \leq D$.

Let $L = \{(a_i, b) : b \in B_i, i \geq \ell\}$. Then d_ℓ is the minimum distance of any element in L. Since the size of L is larger than $\binom{n}{2} - k$, it follows that $\delta \geq d_\ell$. Let i be the index such that $D = D_i$. By Lemma 1.1, we have $D_i \leq (1 + 4/s)d_i$. Therefore, we have

$$D = D_i \leq (1 + 4/s)d_i \leq (1 + 4/s)d_\ell \leq (1 + 4/s)\delta.$$

□

We obtain the value of D (which is an approximation to the k-th smallest distance in S), by computing all values d_i and D_i, $1 \leq i \leq m$. That is, for each point a_i, we compute its nearest and furthest neighbors in the set B_i. We will show how to use the split tree to solve this problem for the case when the points are in $\mathbb{R}^2$. The algorithm uses the following result.

Lemma 1.10 *Let V be a set of N points in the plane. There exists a data structure that supports the following operations:*

1. *For any given query point $q \in \mathbb{R}^2$, report the nearest neighbor of q in V. The query time is $O(\log^2 N)$.*
2. *For any given query point $q \in \mathbb{R}^2$, report the furthest neighbor of q in V. The query time is $O(\log^2 N)$.*
3. *Insert an arbitrary point into the set V. The amortized insertion time is $O(\log^2 N)$.*

Proof: Consider the Voronoi diagram of the set V, which can be constructed in $O(N \log N)$ time. If we store this diagram, together with a point location data structure, then a nearest-neighbor query can be answered in $O(\log N)$ time; see Preparata and Shamos [21] and de Berg *et al.* [11]. If we use the furthest-point Voronoi diagram, then we obtain the same result for furthest-neighbor queries. Unfortunately, these Voronoi diagrams cannot be maintained efficiently under insertions of points. Since nearest-neighbor and furthest-neighbor queries are *decomposable*, however, we can use the *logarithmic method* of Bentley and Saxe [4] to obtain the time bounds that are claimed in the lemma. □

We now show how Lemma 1.10 can be used to compute all values d_i and D_i, $1 \leq i \leq m$. Consider the split tree T. Recall that for any node u, S_u denotes the subset of S that is stored at the leaves in the subtree of u. We store with each node u, a list L_u consisting of all points a_i such that $B_i = S_u$.

The algorithm traverses the split tree T in postorder. During this traversal, the following invariant is maintained: If u is a node that has been traversed, but none of its proper ancestors has been traversed yet, then u stores the data structure DS_u of Lemma 1.10 for the point set S_u.

The postorder traversal of T does the following. Let u be the current node in this traversal. If u is a leaf storing the point, say, p, then we compute $d_i = D_i = |a_i p|$ for each point a_i in L_u, and we build the data structure DS_u of Lemma 1.10 for the singleton set $S_u = \{p\}$. Assume that the current node u is not a leaf.

Let v and w be the two children of u. By the invariant, v and w store the data structure DS_v and DS_w of Lemma 1.10 for the sets S_v and S_w, respectively. Assume without loss of generality that $|S_v| \leq |S_w|$. We do the following: First, we discard the data structure DS_v. Then, we insert each element of S_v into DS_w; this results in the data structure DS_u storing all elements of S_u. Finally, for each element a_i of L_u, we use DS_u to find the nearest and furthest neighbors of a_i in the set S_u, and compute the values of d_i and D_i.

We analyze the total time of this algorithm. Since the WSPD contains $O(n \log n)$ pairs, the total number of nearest-neighbor and furthest-neighbor queries is $O(n \log n)$. Consider any fixed point p of S. If p is inserted into a data structure, then it "moves" to a new set whose size is at least twice the size of the previous set containing p. As a result, p is inserted $O(\log n)$ times. Thus, overall, the total number of insertions is $O(n \log n)$. Lemma 1.10 then implies that the running time of the algorithm is $O(n \log^3 n)$. If we combine this with Theorem 1.2, we obtain the following result.

Theorem 1.11 *Let S be a set of n points in $\mathbb{R}^d$, let k be an integer with $1 \leq k \leq \binom{n}{2}$, let $\epsilon > 0$ be a constant, and let δ be the k-th smallest distance in S. In $O(n \log^3 n)$ time, a pair (x, y) of points in S can be computed for which $\delta \leq |xy| \leq (1 + \epsilon)\delta$.*

1.6 Generalization to metric spaces

All results in the previous sections are valid for Euclidean spaces $\mathbb{R}^d$, where d is a constant. In recent years, the WSPD (and its applications) has been generalized to more general metric spaces.

Consider an arbitrary metric space (S, δ), where S is a set of n elements, and $\delta : S \times S \longrightarrow \mathbb{R}$ is the metric defined on S. For any two subsets A and B of S, we denote the minimum distance between any point in A and any point in B by $\delta(A, B)$, and we denote the diameter of A by $D(A)$. If $s > 0$ is a real number, then we say that A and B are well-separated with respect to s, if

$$\delta(A, B) \geq s \cdot \max(D(A), D(B)).$$

Using this generalized notion of being well-separated, we define a well-separated pair decomposition (WSPD) for S as in Definition 1.2.

Gao and Zhang [14] considered the problem of constructing a WSPD for the unit-disk graph metric: Let

S be a set of n points in $\mathbb{R}^d$. The *unit-disk graph* is defined to be the graph with vertex set S, in which any two distinct points p and q are connected by an edge if and only $|pq| \leq 1$. If we define $\delta(p, q)$ to be the length of a shortest path between p and q in the unit-disk graph, then (S, δ) is a metric space. Observe that even though the unit-disk graph may have $\Theta(n^2)$ edges, it can be represented in $O(n)$ space: It suffices to store the points of S. Given any two points p and q of S, we can decide in $O(1)$ time if p and q are connected by an edge and, if so, compute its length $|pq|$. (If p and q are not connected by an edge, however, then a shortest-path computation is needed to compute $\delta(p, q)$.) Gao and Zhang proved the following result:

Theorem 1.12 *Let S be a set of n points in $\mathbb{R}^d$, and let $s > 1$ be a real number. Consider the unit-disk graph metric on S.*

1. *If $d = 2$, then a WSPD for S with respect to s, consisting of $O(s^4 n \log n)$ pairs, can be computed in $O(s^4 n \log n)$ time.*
2. *If $d = 3$, then a WSPD for S, with respect to s, consisting of $O(n^{4/3})$ pairs, can be computed in $O(n^{4/3} \log^{O(1)} n)$ time.*
3. *If $d \geq 4$, then a WSPD for S, with respect to s, consisting of $O(n^{2-2/d})$ pairs, can be computed in $O(n^{2-2/d})$ time.*

Talwar [27] extended the WSPD to metric spaces whose *doubling dimension* is a constant. To define this notion, let (S, δ) be a metric space. A ball, with center $p \in S$ and radius R, is defined to be the set $\{q \in S : \delta(p, q) \leq R\}$. The *doubling parameter* of S is defined to be the smallest integer λ such that the following holds, for all real numbers $R > 0$: Every ball with radius R can be covered by λ balls of radius $R/2$. The doubling dimension of the metric space is defined to be $\log \lambda$. Observe that this generalizes Euclidean space $\mathbb{R}^d$, because the doubling dimension of $\mathbb{R}^d$ is proportional to d.

Many algorithms solving proximity problems in $\mathbb{R}^d$ are analyzed using a packing argument. If the doubling parameter λ is small, then, in many cases, a similar analysis can be used to efficiently solve these problems.

Talwar showed how to compute a WSPD consisting of $O(s^{\log \lambda} n \log \Delta)$ pairs, where Δ is the aspect ratio, which is defined to be the ratio of the diameter and the closest pair distance. Har-Peled and Mendel [17]

gave an improved construction and obtained the following result:

Theorem 1.13 *Let (S, δ) be a metric space, let $n = |S|$, let λ be the doubling parameter of S, and let $s > 1$ be a real number. There exists a randomized algorithm that constructs, in $O(\lambda n \log n + s^{\log \lambda} n)$ expected time, a well-separated pair decomposition for S, with separation ratio s, consisting of $O(s^{\log \lambda} n)$ pairs.*

Most results of the previous sections remain valid for metric spaces whose doubling parameter is bounded by a constant. Interestingly, Har-Peled and Mendel have shown that this is not the case for the all-nearest neighbors problem: For every deterministic algorithm that solves this problem, there exists a metric space on n points and doubling parameter $\lambda \leq 3$, such that this algorithm must examine all $\binom{n}{2}$ distances determined by these points. This implies that, in such metric spaces, it takes $\Omega(n^2)$ time to compute a minimum spanning tree. Since the greedy spanner contains a minimum spanning tree, this also implies an $\Omega(n^2)$–time lower bound for computing this spanner. For more information, refer to Smid [26].

References

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

[2] M. Aigner and G. M. Ziegler. *Proofs from THE BOOK*. Springer-Verlag, Berlin, 3rd edition, 2004.

[3] I. Althöfer, G. Das, D. P. Dobkin, D. Joseph, and J. Soares. On sparse spanners of weighted graphs. *Discrete & Computational Geometry*, 9:81–100, 1993.

[4] J. L. Bentley and J. B. Saxe. Decomposable searching problems I: Static-to-dynamic transformations. *Journal of Algorithms*, 1:301–358, 1980.

[5] M. Bern and D. Eppstein. Approximation algorithms for geometric problems. In D. S. Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems*, pages 296–345. PWS Publishing Company, Boston, MA, 1997.

[6] S. Bespamyatnikh and M. Segal. Fast algorithms for approximating distances. *Algorithmica*, 33:263–269, 2002.

[7] B. Bollobás and A. Scott. On separating systems. *European Journal of Combinatorics*, 28:1068–1071, 2007.

[8] P. B. Callahan. *Dealing with higher dimensions: the well-separated pair decomposition and its applications*. Ph.D. thesis, Department of Computer Science, Johns Hopkins University, Baltimore, Maryland, 1995.

[9] P. B. Callahan and S. R. Kosaraju. Faster algorithms for some geometric graph problems in higher dimensions. In *Proceedings of the 4th ACM-SIAM Symposium on Discrete Algorithms*, pages 291–300, 1993.

[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. *Journal of the ACM*, 42:67–90, 1995.

[11] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, 3rd edition, 2008.

[12] D. Eppstein. Spanning trees and spanners. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 425–461. Elsevier Science, Amsterdam, 2000.

[13] J. Erickson. On the relative complexities of some geometric problems. In *Proceedings of the 7th Canadian Conference on Computational Geometry*, pages 85–90, 1995.

[14] J. Gao and L. Zhang. Well-separated pair decomposition for the unit-disk graph metric and its applications. *SIAM Journal on Computing*, 35:151–169, 2005.

[15] R. L. Graham and H. O. Pollak. On the addressing problem for loop switching. *Bell System Technical Journal*, 50:2495–2519, 1971.

[16] J. Gudmundsson and C. Knauer. Dilation and detours in geometric networks. In T. F. Gonzalez, editor, *Handbook of Approximation Algorithms and Metaheuristics*. Taylor & Francis, 2nd edition, 2017.

[17] S. Har-Peled and M. Mendel. Fast construction of nets in low-dimensional metrics and their applications. *SIAM Journal on Computing*, 35:1148–1184, 2006.

[18] R. Janardan. On maintaining the width and diameter of a planar point-set online. *International Journal of Computational Geometry & Applications*, 3:331–344, 1993.

[19] M. J. Katz and M. Sharir. An expander-based approach to geometric optimization. *SIAM Journal on Computing*, 26:1384–1408, 1997.

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

[21] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, Berlin, 1988.

[22] E. A. Ramos. An optimal deterministic algorithm for computing the diameter of a three-dimensional point set. *Discrete & Computational Geometry*, 26:233–244, 2001.

[23] J. S. Salowe. Constructing multidimensional spanner graphs. *International Journal of Computational Geometry & Applications*, 1:99–107, 1991.

[24] J. S. Salowe. Enumerating interdistances in space. *International Journal of Computational Geometry & Applications*, 2:49–59, 1992.

[25] M. Smid. Closest-point problems in computational geometry. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 877–935. Elsevier Science, Amsterdam, 2000.

[26] 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.

[27] K. Talwar. Bypassing the embedding: Approximation schemes and compact representations of low dimensional metrics. In *Proceedings of the 36th ACM Symposium on the Theory of Computing*, pages 281–290, 2004.

[28] P. M. Vaidya. Minimum spanning trees in k-dimensional space. *SIAM Journal on Computing*, 17:572–582, 1988.

[29] P. M. Vaidya. An $O(n \log n)$ algorithm for the all-nearest-neighbors problem. *Discrete & Computational Geometry*, 4:101–115, 1989.

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