A Simple Randomized $O(n \log n)$–Time Closest-Pair Algorithm in Doubling Metrics

Anil Maheshwari* Wolfgang Mulzer† Michiel Smid*

April 14, 2020

Abstract

Consider a metric space $(P, dist)$ with N points whose doubling dimension is a constant. We present a simple, randomized, and recursive algorithm that computes, in $O(N \log N)$ expected time, the closest-pair distance in P. To generate recursive calls, we use previous results of Har-Peled and Mendel, and Abam and Har-Peled for computing a sparse annulus that separates the points in a balanced way.

For a long time researchers felt that there might be a quadratic lower bound on the complexity of the closest-pair problem.

— Jon Louis Bentley,
— *Communications of the ACM*, volume 23, page 226, 1980

1 Introduction

The closest-pair problem is one of the oldest problems in computational geometry: Given a set P of N points in the Euclidean space $\mathbb{R}^d$, where $d \geq 1$ is a constant, compute a *closest-pair* in P, i.e., a pair p, q of distinct points in P for which the Euclidean distance $dist(p, q)$ is minimum.

The algorithm of Bentley and Shamos. The first $O(N \log N)$–time algorithm for this problem dates back to 1976 and is due to Bentley and Shamos [5] (See also Bentley [3]). When $d = 2$, the algorithm is particularly simple and an excellent example of a "textbook algorithm" that illustrates the power of the divide-and-conquer paradigm; see Cormen *et al.* [6, Section 33.4] and Kleinberg and Tardos [9, Section 5.4]. Bentley [4] mentions that this algorithm, for $d = 2$, is due to Shamos, and the idea of using divide-and-conquer was suggested by H.R. Strong.

*School of Computer Science, Carleton University, Ottawa, Canada. Research supported by NSERC.

†Institut für Informatik, Freie Universität Berlin, Germany. Research supported in part by ERC StG 757609.

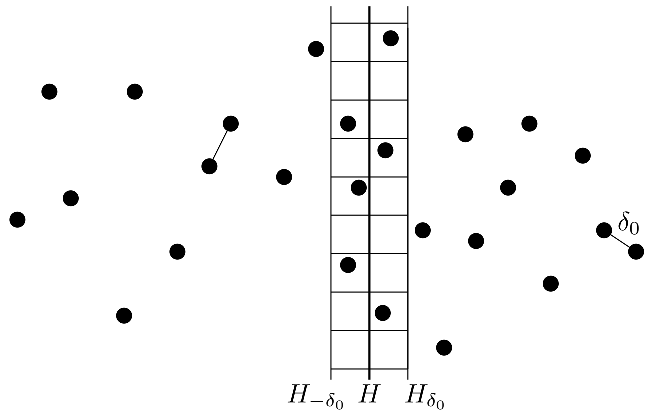

Figure 1: The divide-and-conquer algorithm by Bentley and Shamos in two dimensions: compute a hyperplane H that partitions the point set P evenly; recurse on the "left" and on the "right" part; compute the closest pair in the slab between $H_{-\delta_0}$ and H_{δ_0}, where δ_0 is the minimum of the two closest-pair distances to the left and to the right of H.

We briefly describe the Bentley–Shamos algorithm. In a preprocessing step, for each $i = 1, 2, \ldots, d$, the algorithm sorts the points of P according to their i-th coordinates.

If $d = 1$, the closest-pair in P can easily be computed in $O(N)$ time, by scanning the sorted sequence of elements of P.

Assume that $d \geq 2$. We first introduce some notation. Let H be a hyperplane that is orthogonal to one of the d coordinate axes. For any real number $\delta > 0$, we denote by $H_{-\delta}$ and $H_{+\delta}$ the two hyperplanes that are obtained by translating H by a distance of δ to the "left" and "right", respectively.

Bentley and Shamos prove that there exists a hyperplane H, such that, for some positive constant $\alpha > 0$ that only depends on the dimension d, the following properties hold. First, at least αN points of P are to the "left" of H and at least αN points of P are to the "right" of H. Second, let δ_0 be the smaller of the closest-pair distance to the left of H and the closest-pair distance to the right of H. Then, the *slab* defined by $H_{-\delta_0}$ and $H_{+\delta_0}$ contains $O(N^{1-1/d})$ points of P. Third, such a hyperplane H can be computed in $O(N)$ time. Observe that the exact value of δ_0 is not known when H is computed. However, during the computation of H, we do obtain an upper bound on δ_0.

To compute the closest-pair distance in P, the algorithm recurses on two subproblems in $\mathbb{R}^d$, one subproblem for the points to the left of H and one subproblem for the points to the right of H. Finally, the algorithm must consider the points inside the slab. Observe that these points are "sparse", in the sense that the number of points inside any hypercube with sides of length δ_0 is bounded from above by a function that only depends on d, see Figure 1. Bentley and Shamos use the divide-and-conquer technique to solve the sparse problem, on only $O(N^{1-1/d})$ points, in $O(N)$ time.

The total running time $T(N)$ of this algorithm satisfies the standard merge-sort recurrence

$$T(N) = O(N) + T(N') + T(N''),$$

where $N' \leq (1-\alpha)N$, $N'' \leq (1-\alpha)N$, and $N' + N' = N$. It follows that the algorithm computes the closest-pair distance in P in $O(N \log N)$ time.

Our results. The algorithm of Bentley and Shamos uses the fact that the points in the set P have coordinates. This leads to the problem considered in this paper: Can we compute the closest-pair distance, by only using distances? Thus, we assume that (P, dist) is a metric space (to be defined in Section 2), and we have an oracle that returns, in $O(1)$ time, the distance $\mathit{dist}(p, q)$ for any two elements p and q of P.

In general metric spaces, the closest-pair distance cannot be computed in subquadratic time: Assume that exactly one distance is equal to 1, and all other distances are equal to 2. An easy adversary argument implies that any algorithm that computes the closest-pair distance must take $\Omega(N^2)$ time in the worst case.

In this paper, we present a randomized algorithm that computes the closest-pair distance in $O(N \log N)$ expected time, for the case when the *doubling dimension* of the metric space (P, dist) is bounded by a constant. Informally, this means that any ball can be covered by $O(1)$ balls of half the radius; the formal definition will be given in Section 2.

A closest-pair algorithm can be obtained from results by Har-Peled and Mendel [7]: They show that a well-separated pair decomposition of P can be computed in $O(N \log N)$ expected time. Given this decomposition, the closest-pair distance can be obtained in $O(N)$ time. The drawback of this approach is that this algorithm is quite technical. We show that there is a very simple algorithm that computes the closest-pair distance in $O(N \log N)$ expected time. As we will see later, one of the main ingredients that we use is from [7].

Since the elements of P do not have coordinates, there are no notions of a hyperplane or a slab. It is natural to replace these by a *ball* and an *annulus*; the latter is the subset of points between two concentric balls.

Let d denote the doubling dimension of the metric space (P, dist). Abam and Har-Peled [1], using a previous result of Har-Peled and Mendel [7], show that, in $O(N)$ expected time, two concentric balls of radii R and $R+w$ can be computed, such that, for some positive constant $\alpha > 0$ that only depends on d,

1. the ball of radius R contains at least αN points,
2. there are at least αN points outside the ball of radius $R + w$,
3. the annulus with radii R and $R + w$ contains $O(N^{1-1/d})$ points, and
4. the width w of this annulus is proportional to $R/N^{1/d}$.

We will refer to this annulus as a *sparse separating annulus*, see Figure 2. In Section 3, we will present a simplified version of the algorithm of Abam and Har-Peled [1] that computes such an annulus.

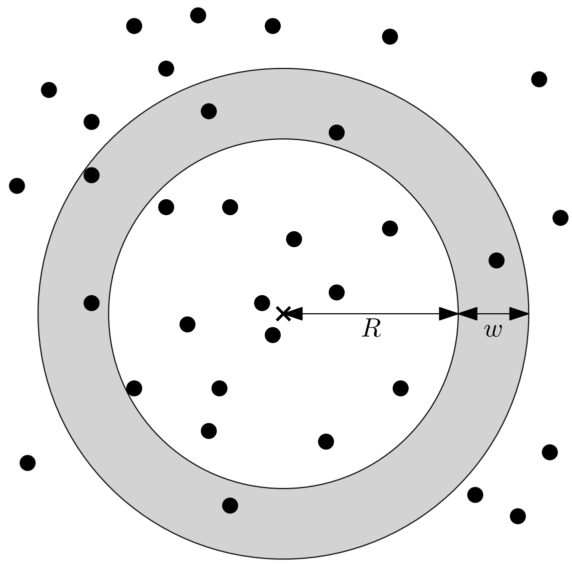

Figure 2: A sparse separating annulus for a planar point set P with N points: Each of the regions inside and outside the annulus contains $\Omega(N)$ points; inside the annulus, there are $O(\sqrt{N})$ points; and the width w of the annulus is proportional to $R/\sqrt{N}$.

Let δ be the closest-pair distance in P. A packing argument (see Section 2.2) shows that the above ball of radius R contains $O((R/\delta)^d)$ points. Since this ball contains at least αN points, it follows that $R = \Omega(\delta \cdot N^{1/d})$. Thus, by choosing appropriate constants, the width w of the above annulus is at least δ. (The formal proofs will be presented in Section 4.) Observe that, as in the Bentley–Shamos algorithm, the value of δ is not known when the two concentric balls are computed.

Let P_1 be the subset of all points that are inside the ball of radius R, let P_2 be the subset of all points that are inside the annulus, and let P_3 be the subset of all points that are outside the ball of radius $R + w$. Then it suffices to recursively run the algorithm twice, once on $P_1 \cup P_2$, and once on $P_2 \cup P_3$. The expected running time of this algorithm satisfies the recurrence,

$$T(N) = O(N) + T(N') + T(N''),$$

where $N' \leq (1-\alpha)N$, $N'' \leq (1-\alpha)N$, and $N' + N'' \leq N + O(N^{1-1/d})$. We will prove in Section 4.2 that this recurrence solves to $T(N) = O(N \log N)$.

2 Metric spaces and their doubling dimension

A *metric space* is a pair (P, dist), where P is a non-empty set and $\mathit{dist} : P \times P \to \mathbb{R}$ is a function such that for all x, y, and z in P,

1. $\mathit{dist}(x,x) = 0$,
2. $\mathit{dist}(x,y) > 0$ if $x \neq y$,

3. $dist(x, y) = dist(y, x)$, and

4. $dist(x, z) \leq dist(x, y) + dist(y, z)$.

The fourth property is called the *triangle inequality*. We refer to $dist(x, y)$ as the *distance* between x and y. We only consider metric spaces in which the set P is finite. We call the elements of P *points*.

If $p \in P$ is a point, $S \subseteq P$ is a subset of P, and R, R' are real numbers with $R' \geq R \geq 0$, then the *ball* in S with *center* p and *radius* R is the set

$$ball_S(p, R) = \{x \in S : dist(p, x) \leq R\},$$

and the *annulus* in S with *center* p, *inner radius* R, and *outer radius* R' is the set

$$annulus_S(p, R, R') = \{x \in S : R < dist(p, x) \leq R'\}.$$

The *closest-pair* distance in S is

$$\delta(S) = \begin{cases} \infty & \text{if } |S| \leq 1, \\ \min\{dist(x, y) : x \in S, y \in S, x \neq y\} & \text{if } |S| \geq 2. \end{cases}$$

The doubling dimension of a metric space was introduced by Assouad [2]; see also Heinonen [8]:

Definition 1 Let $(P, dist)$ be a finite metric space and let λ be the smallest integer such that the following is true: For every point p in P and every real number $R > 0$, $ball_P(p, R)$ can be covered by at most λ balls in P of radius $R/2$. The *doubling dimension* of $(P, dist)$ is defined to be $\log \lambda$.

The doubling dimension is in the interval $[1, \log |P|]$ and, in general, is not an integer. For example, if $dist$ is the Euclidean distance function in $\mathbb{R}^2$, the doubling dimension is $\log 7$, whereas in $\mathbb{R}^d$, the doubling dimension is $\Theta(d)$. The discrete metric space $(P, dist)$ in which the distance between any two distinct points is equal to 1 has doubling dimension $\log |P|$.

2.1 The doubling dimension of a subset

Our algorithm for computing the closest-pair distance in P uses recursion. In a recursive call, the algorithm is run on a subset S of P. We show below that the doubling dimension of S may not be the same as that of P.

Let $(P, dist)$ be a metric space, let d be its doubling dimension, and let S be a non-empty subset of P. To determine the doubling dimension of $(S, dist)$,[1] we have to cover any ball $ball_S(p, R)$, with $p \in S$ and $R > 0$, by balls in S of radius $R/2$ that are centered at points of S. The number of such balls may be larger than 2^d.

[1] With a slight abuse of notation, when writing $(S, dist)$, we consider $dist$ to be the restriction of the distance function to the set $S \times S$.

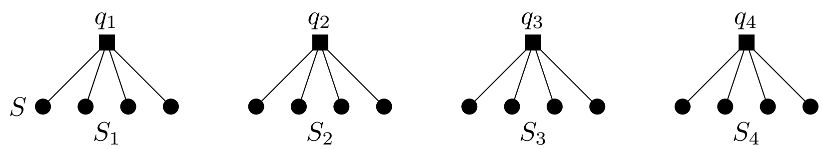

Figure 3: A metric space P of 20 points and a subset $S = \bigcup_{i=1}^{4} S_i$ of P with strictly smaller doubling dimension. For $i = 1, \ldots, 4$, the distance between q_i and all points in S_i is 1. All other distances between pairs of distinct points are 2.

To give an example, let n be a positive integer and let (S, dist) be the metric space of size n^2 with $\mathit{dist}(x, y) = 2$ for all distinct points x and y in S. The doubling dimension d_S of (S, dist) is equal to

$$d_S = \log |S| = 2 \log n.$$

Partition S into subsets $S_1, S_2, \ldots, S_n$, each consisting of n points. Let $q_1, q_2, \ldots, q_n$ be new points, and let

$$P = S \cup \{q_1, q_2, \ldots, q_n\}.$$

(For an illustration with $n = 4$, refer to Figure 3.) For any two points x and y in P, define

$$\mathit{dist}(x, y) = \begin{cases} 0 & \text{if } x = y, \\ 1 & \text{if there is an } i \text{ such that } x = q_i \text{ and } y \in S_i, \text{ or } x \in S_i \text{ and } y = q_i, \\ 2 & \text{otherwise.} \end{cases}$$

Since all distances between distinct points are 1 or 2, it follows that (P, dist) fulfills the triangle inequality. Hence, (P, dist) is a metric space. We will prove below that the doubling dimension d of this metric space is equal to

$$d = \log(n + 1).$$

Thus, for large values of n, the ratio d_S/d converges to 2.

To determine the doubling dimension of (P, dist), let p be a point of P, let $R > 0$ be a real number, and let $B = \mathit{ball}_P(p, R)$. If $R \in (0, 1)$, then B is a singleton set, which is covered by the ball $\mathit{ball}_P(p, R/2)$. If $R \in [2, \infty)$, then $B = P$, which is covered by the n balls in P of radius $R/2$ that are centered at $q_1, q_2, \ldots, q_n$. If $R \in [1, 2)$, then $B = \{q_i\} \cup S_i$ for some i. In this case, B can only be covered by the $n + 1$ balls in P of radius $R/2$ that are centered at the points of B. Thus, for each case, we have shown that B can be covered by at most $n + 1$ balls in P of radius $R/2$, and for some B, we need $n + 1$ such balls. This proves that $d = \log(n + 1)$.

The following lemma states that the doubling dimension of a subset S of P is always at most twice the doubling dimension of P.

Lemma 1 *Let (P, dist) be a metric space, let d be its doubling dimension, and let S be a non-empty subset of P. Then the metric space (S, dist) has doubling dimension at most $2d$.*

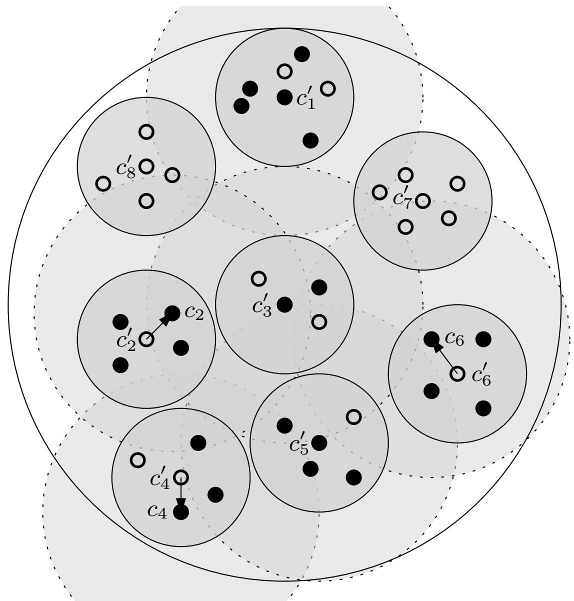

Figure 4: Illustration of the proof of Lemma 1. The points of S are solid; the points of $P \setminus S$ are empty. The ball B' can be covered in P by 8 balls $B'_1, \ldots, B'_8$ with centers $c'_1, \ldots, c'_8$. For $B'_1, \ldots, B'_6$, the intersection with S is nonempty. The centers c'_1, c'_3, and c'_5 are also in S, the centers c'_2, c'_5, and c'_6 must be moved. This increases the covering radius to $R/2$.

Proof. Let p be a point in S, let $R > 0$ be a real number, and consider the ball $B = ball_S(p, R)$ in S. Let $B' = ball_P(p, R)$ be the corresponding ball in P. By applying the definition of doubling dimension twice, we can cover B' by balls B'_i, for $1 \leq i \leq 2^{2d}$, in P, each having radius $R/4$. Let k be the number of indices i for which $B'_i \cap S \neq \emptyset$. We may assume, without loss of generality, that $B'_i \cap S \neq \emptyset$ for all i with $1 \leq i \leq k$, and $B'_i \cap S = \emptyset$ for all i with $k + 1 \leq i \leq 2^{2d}$. For $i = 1, 2, \ldots, k$, let $c'_i \in P$ be the center of B'_i, let

$$c_i = \begin{cases} c'_i & \text{if } c'_i \in S, \\ \text{an arbitrary point in } B'_i \cap S & \text{if } c'_i \notin S, \end{cases}$$

and let $B_i = ball_S(c_i, R/2)$, see Figure 4.

We claim that the balls B_i in S, $1 \leq i \leq k$, cover the ball B. To prove this, let q be a point in B. Then, $q \in B'$ and, thus, there is an index i, $1 \leq i \leq k$, with $q \in B'_i$. Since

$$dist(c_i, q) \leq dist(c_i, c'_i) + dist(c'_i, q) \leq R/4 + R/4 = R/2,$$

the point q is in the ball B_i. We have shown that any ball in S of radius R can be covered by at most 2^{2d} balls in S of radius $R/2$. ∎

2.2 The packing lemma

Consider a metric space $(P, dist)$ whose doubling dimension is "small", and a ball B in P whose radius R is proportional to the closest-pair distance $\delta(P)$. By repeatedly applying the definition of doubling dimension, we can cover B by a "small" number of balls of radius less than $\delta(P)$. Since each of these smaller balls contains only one point, the original ball B cannot contain "many" points. The following lemma formalizes this.

Lemma 2 *Let $(P, dist)$ be a finite metric space with $|P| \geq 2$ and doubling dimension d. Let δ be the closest-pair distance in P. Then, for any point p in P and any real number $R \geq \delta/2$,*

$$|ball_P(p, R)| \leq (4R/\delta)^d.$$

Proof. Set $k = \lceil \log(2R/\delta) \rceil$. Then, $k \geq 0$ and $2R/\delta \leq 2^k < 4R/\delta$. We apply the definition of doubling dimension k times in order to cover $ball_P(p, R)$ by $2^{kd} \leq (4R/\delta)^d$ balls of radius $R/2^k < \delta$. Each of these 2^{kd} balls contains exactly one point of P, namely its center. ∎

3 Computing a sparse separating annulus

Throughout this section, $(P, dist)$ is a finite metric space, d denotes its doubling dimension, S is a non-empty subset of P, and n denotes the size of S. Observe that d will always refer to the doubling dimension of the entire metric space $(P, dist)$.

In this section, we present a simplified variant of the algorithm of Abam and Har-Peled [1] to compute the sparse separating annulus that was mentioned in Section 1.

3.1 Computing a separating annulus

Let $\mu \geq 1$ be a real number (possibly depending on n) and set $c = 2(8\mu)^d$. Assume that $n \geq c + 1$. As a first step, we give a randomized algorithm that computes a point p in S and a real number $R' > 0$, such that $|ball_S(p, R')| \geq n/c$ and $|ball_S(p, \mu R')| \leq n/2$. This algorithm is due to Har-Peled and Mendel [7, Lemma 2.4]; see also Abam and Har-Peled [1, Lemma 2.6]. In order to be self-contained, we present the algorithm and its analysis.

The algorithm chooses a uniformly random point p in S and computes the smallest radius R_p such that $ball_S(p, R_p)$ contains at least n/c points. Then it checks if $ball_S(p, \mu R_p)$ contains at most $n/2$ points. If this is the case, the algorithm returns p and R_p. Otherwise, the algorithm is repeated. The pseudocode for this algorithm is given below.

Algorithm SepAnn(S, n, d, μ, c)

Comment: The input is a subset S, of size n, of a metric space of doubling dimension d, and real numbers $\mu \geq 1$ and $c > 1$. If $c = 2(8\mu)^d$ and $n \geq c+1$, then the algorithm returns a point p in S and a real number $R' > 0$ that satisfy the two properties in Lemma 3.

repeat p = uniformly random point in S;
    $R_p = \min\{r > 0 : |ball_S(p, r)| \geq n/c\}$
until $|ball_S(p, \mu R_p)| \leq n/2$;
$R' = R_p$;
return p and R'

Lemma 3 *Let $\mu \geq 1$ be a real number (possibly depending on n) and set $c = 2(8\mu)^d$. Assume that $n \geq c+1$. Algorithm* SepAnn(S, n, d, μ, c) *has expected running time $O(cn)$. It returns a point p in S and a real number $R' > 0$, such that*

1. $|ball_S(p, R')| \geq n/c$ *and*
2. $|ball_S(p, \mu R')| \leq n/2$.

Proof. Let p be a point in S. Since $n \geq c+1$, we have $|ball_S(p, R_p)| \geq n/c > 1$. Therefore, $ball_S(p, R_p)$ contains at least two points of S, which means that $R_p > 0$. The radius R_p can be found in $O(n)$ time, by selecting the $\lceil n/c \rceil$-th smallest element in the sequence of distances between p and all points of S (including p itself). By scanning this sequence, we can compute $|ball_S(p, \mu R_p)|$ in $O(n)$ time. Thus, one iteration of the algorithm takes $O(n)$ time.

We say that a point p in S is *good*, if $|ball_S(p, \mu R_p)| \leq n/2$. We will prove below that a uniformly random point of S is good with probability at least $1/c$. This will imply that the expected number of iterations of the algorithm is at most c and, therefore, the expected running time is $O(cn)$.

Consider a ball in P of minimum radius that contains at least n/c points of S and that is centered at a point of P. Let $q \in P$ be the center of this ball and let R be its radius. We claim that every point in $ball_S(q, R)$ is good. This will imply that a uniformly random point in S has probability at least $1/c$ of being good. See Figure 5 for an illustration of the argument.

To prove the claim, let p be a point in $ball_S(q, R)$. We will show that $|ball_S(p, \mu R_p)| \leq n/2$. We first observe that

$$ball_S(q, R) \subseteq ball_S(p, 2R).$$

Indeed, if $x \in ball_S(q, R)$, then

$$dist(p, x) \leq dist(p, q) + dist(q, x) \leq R + R = 2R$$

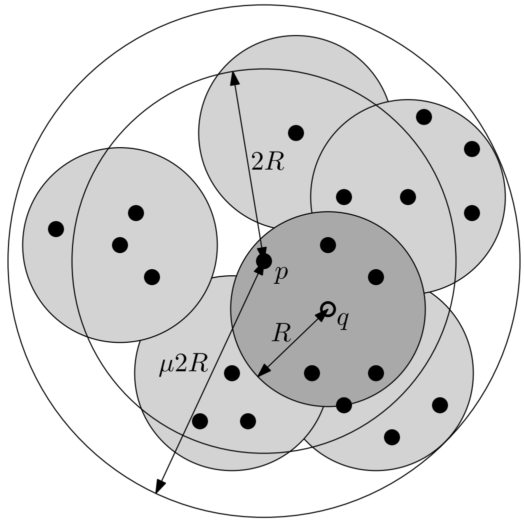

Figure 5: Illustration of the proof of Lemma 3. The point $q \in P$ and the radius R are such that $ball_P(q, R)$ is the minimum-radius ball in P that contains at least n/c points from S. If we pick an arbitrary point $p \in ball_S(q, R)$, then $ball_S(p, 2R)$ covers $ball_S(q, R)$ and hence contains at least n/c points. The ball $ball_P(p, \mu 2R)$ can be covered by $c/2$ balls in P of radius R, and hence it contains at most $n/2$ points from S.

---

and, therefore, $x \in ball_S(p, 2R)$. It follows that

$$|ball_S(p, 2R)| \geq |ball_S(q, R)| \geq n/c,$$

which implies that

$$R_p \leq 2R.$$

Let $k = \lceil \log(4\mu) \rceil$. By the definition of doubling dimension, we can cover $ball_P(p, \mu 2R)$ by $2^{kd} < 2^{(\log(4\mu)+1)d} = (8\mu)^d = c/2$ balls in P of radius $\mu 2R/2^k < R$. By the definition of R, each of these (at most) $c/2$ balls contains less than n/c points of S. Therefore,

$$|ball_S(p, \mu R_p)| \leq |ball_S(p, \mu 2R)| < c/2 \cdot n/c = n/2.$$

Thus, we have shown that every point in $ball_S(q, R)$ is good. ∎

Remark 1 Consider the parameters μ, c, and k in Lemma 3 and its proof. If $\log(4\mu)$ is not an integer, then we can take $k = \lceil \log(2\mu) \rceil$ and reduce the value of c to $2(4\mu)^d$.

3.2 A refinement of the algorithm

Algorithm SepAnn(S, n, d, μ, c) returns a point p and a real number $R' > 0$, such that $|ball_S(p, R')| \geq n/c$ and $|ball_S(p, \mu R')| \leq n/2$. The annulus $annulus_S(p, R', \mu R')$ may contain $\Theta(n)$ points. In this section, we present a refinement of this algorithm that outputs an

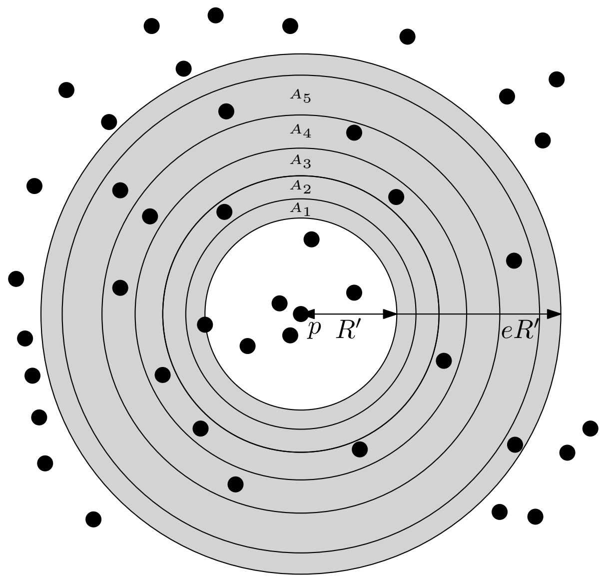

Figure 6: The annulus $annulus_S(p, R', eR')$ contains $t = 5$ annuli $A_1, A_2, \ldots, A_5$. At least one of them contains at most $n/10$ points, and at least 3 of them contain at most $n/5$ points.

---

annulus that contains a "small" number of points of S. Our algorithm is a simplified version of an algorithm due to Abam and Har-Peled [1, Lemma 2.7].

The refined algorithm takes as input an integer $t \geq 1$ that may depend on n. First it runs algorithm SepAnn(S, n, d, μ, c) with $\mu = e$ and (by Remark 1) $c = 2(4e)^d$. Consider the output p and R'. Recall that (since $n \geq c + 1$) we have $R' > 0$. Let

$$R_i = (1 + 1/t)^i \cdot R'$$

for $i = 0, 1, \ldots, t$, and

$$A_i = annulus_S(p, R_{i-1}, R_i)$$

for $i = 1, 2, \ldots, t$. The inequality $1 + x \leq e^x$, which is valid for all real numbers x, implies that, for each i with $0 \leq i \leq t$,

$$R_i \leq \left(e^{1/t}\right)^i \cdot R' = e^{i/t} \cdot R' \leq eR'.$$

Thus, the t annuli A_i are contained in $annulus_S(p, R', eR')$, see Figure 6. Observe that they are pairwise disjoint and, together, contain at most $n/2$ points of S. Therefore, there is an i such that $|A_i| \leq n/(2t)$. We can compute $|A_1|, |A_2|, \ldots, |A_t|$ and, thus, the smallest of these values, as follows: Any point x in S with $R' = R_0 < dist(p, x) \leq R_t$ is contained in A_j, where

$$j = \left\lceil \frac{\log(dist(p, x)/R')}{\log(1 + 1/t)} \right\rceil .$$

Thus, by scanning the sequence of distances between p and all points of S, we can compute, in $O(n)$ time, an index i such that $|A_i| \leq n/(2t)$. This is the approach of Abam and Har-Peled [1].

Our simplification uses the fact that, on average, one annulus A_i contains at most $n/(2t)$ points of S and, thus, by Markov's inequality, at least $t/2$ of these annuli contain at most n/t points of S. The algorithm finds such an annulus A_i by repeatedly choosing a uniformly random element i from $\{1, 2, \ldots, t\}$. As soon as $|A_i| \leq n/t$, the algorithm returns p and R_{i-1}. The pseudocode for this algorithm is given below.

Algorithm SPARSESEPANN(S, n, d, t)

Comment: The input is a subset S, of size $n \geq 2(4e)^d + 1$, of a metric space of doubling dimension d, and an integer $t \geq 1$. The algorithm returns a point p in S and a real number $R > 0$ that satisfy the three properties in Lemma 4.

$c = 2(4e)^d$;
let $p \in S$ and $R' > 0$ be the output of algorithm SEPANN(S, n, d, e, c);
repeat $i =$ uniformly random element in $\{1, 2, \ldots, t\}$;
    $s = |A_i|$
until $s \leq n/t$;
$R = R_{i-1}$;
return p and R

Lemma 4 *Let $t \geq 1$ be an integer (possibly depending on n) and let $c = 2(4e)^d$. Assume that $n \geq c + 1$. Algorithm* SPARSESEPANN(S, n, d, t) *has expected running time $O(cn)$. It returns a point p in S and a real number $R > 0$, such that*

1. $|ball_S(p, R)| \geq n/c$,
2. $|annulus_S(p, R, (1 + 1/t)R)| \leq n/t$, *and*
3. $|S \setminus ball_S(p, (1 + 1/t)R)| \geq n/2$.

Proof. Consider the output p and R' of algorithm SEPANN(S, n, d, e, c). We have seen above that the annuli $A_1, A_2, \ldots, A_t$ are contained in $ball_S(p, eR')$ and, thus, together, contain at most $n/2$ points of S. Moreover, at least $t/2$ of these annuli contain at most n/t points of S. Therefore, in one iteration of the repeat-until-loop in algorithm SPARSESEPANN(S, n, d, t), the size of A_i is at most n/t with probability at least $1/2$. It follows that the expected number of iterations of this repeat-until-loop is at most two. Since one iteration takes $O(n)$ time (by scanning the sequence of distances between p and all points of S), the entire repeat-until-loop takes expected time $O(n)$. This, together with Lemma 3, implies that the expected running time of algorithm SPARSESEPANN(S, n, d, t) is $O(cn)$.

Consider the output p and $R = R_{i-1}$. We have

$$|ball_S(p, R)| \geq |ball_S(p, R')| \geq n/c$$

and

$$|annulus_S(p, R, (1 + 1/t)R)| = |A_i| \leq n/t,$$

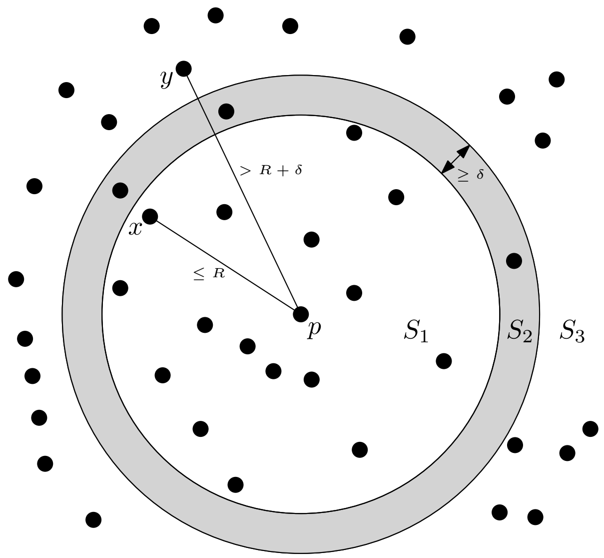

Figure 7: The recursion of the closest pair algorithm. The annulus $annulus_S(p, R, (1+1/t)R)$ splits S into three point sets S_1, S_2, and S_3. It has width at least δ and contains $O(n^{1-1/d})$ points. The distance between any point x in S_1 and any point y in S_3 is more than δ.

proving the first two properties in the lemma. Since

$$|ball_S(p, (1 + 1/t)R)| = |ball(p, R_i)| \leq |ball(p, eR')| \leq n/2,$$

we have

$$|S \setminus ball_S(p, (1 + 1/t)R)| \geq n/2,$$

proving the third property in the lemma. ∎

4 The closest-pair algorithm

Let $(P, dist)$ be a finite metric space, let $N = |P|$, let d be its doubling dimension, and let δ be its closest-pair distance. The recursive algorithm ClosestPair(P, N, d) returns the value of δ. In a generic call, the algorithm takes a subset S of P as input and returns a value δ_0 that is at least δ. If the closest-pair distance in S is equal to δ, then $\delta_0 = \delta$. As before, in each recursive call, d refers to the doubling dimension of the entire metric space $(P, dist)$.

4.1 The algorithm

Let S be a subset of P and let $n = |S|$. If n is small, then algorithm ClosestPair(S, n, d) computes the closest-pair distance in S by brute force. Otherwise, the algorithm runs SparseSepAnn(S, n, d, t), where t is proportional to $n^{1/d}$. Consider the output $p \in S$ and $R > 0$. By Lemmas 2 and 4, $annulus_S(p, R, (1 + 1/t)R)$ contains at most $n/t = O(n^{1-1/d})$

points of S and its width is at least (the unknown value of) δ, see Figure 7. Therefore, it suffices to generate two recursive calls, one on the points in $ball_S(p, (1 + 1/t)R)$ and one on the points outside $ball_S(p, R)$. The pseudocode is given below.

Algorithm ClosestPair(S, n, d)

Comment: The input is a subset S, of size $n \geq 2$, of the metric space $(P, dist)$ of doubling dimension d. The algorithm returns a real number δ_0 that satisfies the two properties in Lemma 5.

```
if n < 2(16e)^d
then compute the closest-pair distance δ0 in S by brute force
else t = ⌊ (1/16e) (n/2)^{1/d} ⌋;
    let p ∈ S and R > 0 be the output of algorithm SparseSepAnn(S, n, d, t);
    S1 = ball_S(p, R);
    S2 = annulus_S(p, R, (1 + 1/t)R);
    S3 = S \ (S1 ∪ S2);
    n' = |S1 ∪ S2|;
    n'' = |S2 ∪ S3|;
    δ' = ClosestPair(S1 ∪ S2, n', d);
    δ'' = ClosestPair(S2 ∪ S3, n'', d);
    δ0 = min(δ', δ'')
endif;
return δ0
```

Recall that $c = 2(4e)^d$ in algorithm SparseSepAnn(S, n, d, t). Therefore,

$$t = \left\lfloor \frac{1}{4}(n/c)^{1/d} \right\rfloor . \tag{1}$$

Before we prove the correctness of algorithm ClosestPair, we show that it terminates. Assume that $n \geq 2(16e)^d$. Then, $n \geq c + 1$ and, by Lemma 4, $|S_1| \geq 2$ and $|S_3| \geq 2$. It follows that both n' and n'' are at most $n - 2$ and, thus, both recursive calls are on sets of sizes less than n.

Lemma 5 *Let δ be the closest-pair distance in P, let S be a subset of P, let $n \geq 2$ be the size of S, and let δ_0 be the output of algorithm* ClosestPair(S, n, d). *Then,*

1. $\delta_0 \geq \delta$ *and*
2. *if* $\delta(S) = \delta$, *then* $\delta_0 = \delta$.

Proof. The first claim holds, because the output δ_0 is always the distance between some pair of distinct points in S. We prove the second claim by induction on n. This second claim obviously holds if $2 \leq n < 2(16e)^d$. Assume that $n \geq 2(16e)^d = 4^d c$ and $\delta(S) = \delta$. Moreover, assume that the second claim holds for all subsets of S containing at least two and less than n points. Observe that $t \geq 1$.

Consider the output $p \in S$ and $R > 0$ of algorithm SparseSepAnn(S, n, d, t). By Lemma 4, $|ball_S(p, R)| \geq n/c > 1$, which implies that $ball_S(p, R)$ contains at least two points (with p being one of them). It follows that $R \geq \delta$. Thus, by Lemma 2,

$$|ball_S(p, R)| \leq |ball_P(p, R)| \leq (4R/\delta)^d.$$

By combining the two inequalities on $|ball_S(p, R)|$, we get

$$n/c \leq (4R/\delta)^d,$$

which, using (1), implies that

$$R \geq (\delta/4) \cdot (n/c)^{1/d} \geq \delta t.$$

The width of $annulus_S(p, R, (1+1/t)R)$ is equal to R/t, which is at least $\delta = \delta(S)$. It follows that the closest-pair distance in S cannot be between one point in S_1 and one point in S_3. To prove this, let x be a point in S_1 and let y be a point in S_3. Then $dist(p, x) \leq R$ and $dist(p, y) > (1 + 1/t)R$, see Figure 7. Thus,

$$(1 + 1/t)R < dist(p, y) \leq dist(p, x) + dist(x, y) \leq R + dist(x, y),$$

which implies $dist(x, y) > R/t \geq \delta = \delta(S)$. It follows that the closest-pair distance in S is within the set $S_1 \cup S_2$ or within the set $S_2 \cup S_3$. By the first claim in the lemma, both δ' and δ'' are at least δ. By the induction hypothesis, at least one of δ' and δ'' is equal to δ. ■

Lemma 5, with $S = P$, proves that algorithm ClosestPair(P, N, d) returns the closest-pair distance in the set P:

Corollary 1 *Let $(P, dist)$ be a metric space of size $N \geq 2$, and let d be its doubling dimension. The output of algorithm* ClosestPair(P, N, d) *is the closest-pair distance in P.*

It remains to analyze the expected running time of the algorithm. For any integer $n \geq 2$, let $T(n)$ denote the maximum expected running time of algorithm ClosestPair(S, n, d), on any subset S of P of size n. Below, we derive a recurrence for $T(n)$.

Assume that $n \geq 2(16e)^d = 4^d c$. Consider the sets S_1, S_2, and S_3 that are computed in the call to ClosestPair(S, n, d). By Lemma 4, $|S_1| \geq n/c$, $|S_2| \leq n/t$, and $|S_3| \geq n/2$. Thus, the values of $n' = |S_1 \cup S_2|$ and $n'' = |S_2 \cup S_3|$ satisfy

$$2 \leq n' \leq (1 - 1/c)n, \tag{2}$$

$$2 \leq n'' \leq (1 - 1/c)n, \tag{3}$$

and

$$n' + n'' \leq n + n/t. \tag{4}$$

Observe that even though n' and n'' are random variables, their values always satisfy (2)–(4).

By Lemma 4, the expected running time of algorithm ClosestPair(S, n, d) is equal to the sum of $O(cn)$ and the total expected times for the two recursive calls. We assume for simplicity that the constant in $O(cn)$ is equal to 1. Thus, we have

$$T(n) \leq cn + \max_{n',n''} \left(T(n') + T(n'')\right), \tag{5}$$

where the maximum ranges over all n' and n'' that satisfy (2)–(4).

If we replace (4) by $n' + n'' \leq n$, then (5) is the standard merge-sort recurrence, whose solution is $O(n \log n)$. In Section 4.2, we will prove that, even with (4), $T(n) = O(n \log n)$, where the constant factor depends only on the doubling dimension of P. This will prove the main result of this paper:

Theorem 1 *Let $(P, dist)$ be a metric space of size $N \geq 2$, and let d be its doubling dimension. Assume that d does not depend on N. The closest-pair distance in P can be computed in $O(N \log N)$ expected time. The constant factor in this time bound depends only on d.*

4.2 Solving the recurrence

Throughout this section, we assume for simplicity that d is an integer. (If this is not the case, then we replace d by $\lceil d \rceil$.) Before we turn to the recurrence (5), we derive some inequalities that will be used later.

Recall the definition of t, see (1). If $n \geq 2(32e)^d = 8^d c$, then

$$t = \left\lfloor \frac{1}{4}(n/c)^{1/d} \right\rfloor \geq \frac{1}{4}(n/c)^{1/d} - 1 \geq \frac{1}{8}(n/c)^{1/d},$$

which implies that

$$n/t \leq 8c^{1/d} n^{1-1/d}. \tag{6}$$

Since

$$\lim_{n \to \infty} \frac{n}{\ln^d n} = \infty,$$

there exists an N_0 such that for all $n \geq N_0$,

$$n \geq 16^d c^{d+1} \ln^d n. \tag{7}$$

We claim that $N_0 = e^{\alpha(d+1)!}$, where $\alpha = 16^d c^{d+1}$, has this property. To prove this, let $m \geq \alpha(d+1)!$. Then

$$e^m = \sum_{k=0}^{\infty} \frac{m^k}{k!} \geq \frac{m^{d+1}}{(d+1)!} \geq \alpha m^d$$

and, thus, if $n \geq N_0$,

$$n = e^{\ln n} \geq \alpha \ln^d n.$$

Define A to be the maximum of $2c^2$ and

$$\max\left\{\frac{T(k)}{k\ln k} : 2 \leq k < N_0\right\}.$$

Observe that A only depends on d.

We will prove that for all integers n with $2 \leq n \leq N$,

$$T(n) \leq An\ln n. \qquad (8)$$

The proof is by induction on n. If $2 \leq n < N_0$, then (8) follows from the definition of A.

Let $n \geq N_0$, and assume that (8) holds for all values less than n. Let n' and n'' be two integers that satisfy (2)–(4). By the induction hypothesis, we have

$$T(n') \leq An'\ln n' \leq An'\ln((1-1/c)n)$$

and

$$T(n'') \leq An''\ln n'' \leq An''\ln((1-1/c)n),$$

implying that

$$T(n') + T(n'') \leq A(n'+n'')\ln((1-1/c)n) \leq A(n+n/t)\ln((1-1/c)n).$$

From (6), we get

$$\begin{aligned} T(n') + T(n'') &\leq A\left(n + 8c^{1/d}n^{1-1/d}\right)\ln((1-1/c)n) \\ &= An\ln n + An\ln(1-1/c) + 8Ac^{1/d}n^{1-1/d}\ln((1-1/c)n) \\ &\leq An\ln n + An\ln(1-1/c) + 8Ac^{1/d}n^{1-1/d}\ln n \\ &\leq An\ln n - An/c + 8Ac^{1/d}n^{1-1/d}\ln n, \end{aligned}$$

where in the last step we used the inequality $\ln(1-x) \leq -x$, which is valid for all real numbers x with $x < 1$. By the definition of A, we have $A \geq 2c^2$, implying that

$$A/c - c \geq A/(2c).$$

Thus,

$$cn + T(n') + T(n'') \leq An\ln n - An/(2c) + 8Ac^{1/d}n^{1-1/d}\ln n.$$

By (7), we have

$$n^{1/d} \geq 16c^{1+1/d}\ln n$$

and, therefore,

$$8Ac^{1/d}n^{1-1/d}\ln n \leq An/(2c).$$

We conclude that

$$cn + T(n') + T(n'') \leq An\ln n.$$

Since n' and n'' were arbitrary integers satisfying (2)–(4), we have shown that (8) holds for the current value of n. Thus, (8) holds for all integers n with $2 \leq n \leq N$.

5 Concluding remarks

We have presented a very simple randomized algorithm for computing the closest-pair distance in metric spaces of small doubling dimension. The algorithm only uses the following operations:

1. For any given point p, count or determine all points that are within a given distance from p, or within a given range of distances from p. This operation can obviously be done in linear time, by simply scanning the sequence of distances between p and all points.

2. For a given sequence of n real numbers, find the k-th smallest element in this sequence. This operation can be done in expected linear time, again by a simple randomized algorithm; see Cormen *et al.* [6, Chapter 9] and Kleinberg and Tardos [9, Section 13.5].

Acknowledgements

This research was carried out at the *Eighth Annual Workshop on Geometry and Graphs*, held at the Bellairs Research Institute in Barbados, January 31 – February 7, 2020. The authors are grateful to the organizers and to the participants of this workshop.

References

[1] M. A. Abam and S. Har-Peled. New constructions of SSPDs and their applications. *Computational Geometry: Theory and Applications*, 45:200–214, 2012.

[2] P. Assouad. Plongements lipschitziens dans $\mathbb{R}^N$. *Bulletin de la Société Mathématique de France*, 111:429–448, 1983.

[3] J. L. Bentley. *Divide and Conquer Algorithms for Closest Point Problems in Multidimensional Space.* Ph.D. thesis, Department of Computer Science, University of North Carolina, Chapel Hill, N.C., 1976.

[4] J. L. Bentley. Multidimensional divide-and-conquer. *Communications of the ACM*, 23:214–229, 1980.

[5] J. L. Bentley and M. I. Shamos. Divide-and-conquer in multidimensional space. In *Proceedings of the 8th ACM Symposium on the Theory of Computing*, pages 220–230, 1976.

[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms.* MIT Press, 3rd edition, 2009.

[7] 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.

[8] J. Heinonen. *Lectures on Analysis on Metric Spaces*. Springer-Verlag, 2001.

[9] J. Kleinberg and E. Tardos. *Algorithm Design*. Addison-Wesley, 2006.