Fast Algorithms for Approximating Distances

Sergei Bespamyatnikh* and Michael Segal†

Department of Computer Science
University of British Columbia
Vancouver, B.C. Canada V6T 1Z4

Abstract

In this paper we present efficient approximate algorithms for the various distance problems. Our technique is based on the well-separated pair decomposition proposed in [6].

1 Introduction

Let S be a set of n points in the plane and let $1 \leq k \leq \frac{n(n-1)}{2}$. Let $d_1 \leq d_2 \leq \ldots \leq d_{\binom{n}{2}}$ be the L_p-distances determined by the pairs of points in S. In this paper we consider the following optimization problems:

- **Distance selection.** Compute the k-th smallest distance between a pair of points of S.
- **Reporting distances.** Report or enumerate the k smallest (largest) distances determined by pairs of points of S in increasing (decreasing) order.
- **Distance ranking.** Determine how many pair points are closer than unit distance apart.
- **Distance by query.** Preprocess a set S into a data structure, so that, given a number k as above, one can answer efficiently what is the k-th smallest distance between a pair of points of S.
- **Reporting bichromatic distances.** Given sets R and B of n red and m blue points, respectively, in the plane and given k, $1 \leq k \leq nm$ report or enumerate k smallest (largest) distances determined by (red-blue) pairs of points.

The problems above received a lot of attention during the past decade. The solution to the distance selection problem can be obtained using a parametric searching. The decision problem is to compute, for a given real r, the sum $\Sigma_{p \in S}|D_r(p) \cap (S - \{p\})|$, where $D_r(p)$ is the closed disk of radius r centered at p. By solving the decision problem we solve our distance ranking problem [1, 11]. Agarwal et al. [1] gave an $O(n^{\frac{4}{3}} \log^{\frac{4}{3}} n)$ expected-time randomized algorithm for the decision problem, which yields an $O(n^{\frac{4}{3}} \log^{\frac{8}{3}} n)$ expected-time algorithm for the distance selection problem. Goodrich [12] derandomized this algorithm, at a cost of an additional polylogarithmic factor in

*Work by Sergei Bespamyatnikh has been supported by a NSERC grant. E-mail: besp@cs.ubc.ca

†Work by Michael Segal has been supported by the Pacific Institute for Mathematical Studies and by a NSERC grant. Email: msegal@cs.ubc.ca

the runtime. Katz and Sharir [13] obtained an expander-based $O(n^{4/3}\log^{3+\varepsilon} n)$-time deterministic algorithm for this problem. By applying a randomized approach Chan [7] was able to obtain an $O(n\log n + n^{2/3}k^{1/3}\log^{5/3} n)$ expected time algorithm for this problem. For our best knowledge, nothing has been done for the query version of the distance selection problem. One may find it useful in parametric searching applications, where a set of candidate solutions is defined by the distances between pairs of points of S, see e.g. [2].

The distance selection problem is also closely related to the reporting distances problem. The papers of Dickerson and Shugart [10], Katoh and Iwano [14] and Segal and Kedem [20] present several algorithms for the enumerating largest k distances. The algorithm in [10] works for any metric, and requires $O(n+k)$ space with expected runtime of $O(n\log n + \frac{k\log k\log n}{\log\log n})$. The paper of Katoh and Iwano [14] presents an algorithm for the L_2 metric with running time $O(\min(n^2, n\log n + k^{4/3}\log n/(\log k)^{1/3}))$ and space $O(n + k^{4/3}/(\log k)^{1/3} + k\log n)$. Their algorithm is based on the k nearest neighbor Voronoi diagrams. The algorithm proposed in [20] is based on posets, works for rectilinear distances and has $O(n + k\log n)$ running time and linear space. Dickerson et al. [9] present an algorithm for the enumeration of all the k smallest distances in S in increasing order. Their algorithm works in time $O(n\log n + k\log k)$ and uses $O(n+k)$ space. Lenhof et al. [15], Salowe [19], Dickerson and Eppstein [8] solve the same problem too but they just report the k closest pairs of points without sorting the distances, spending $O(n\log n + k)$ time and $O(n+k)$ space. An algorithm for solving the enumerating problem (for the smallest k distances) is also presented in [8], spending $O(n\log n + k\log k)$ time and using $O(n+k)$ space. Chan [7] presents $O(n\log n + k)$ expected time simple algorithm for reporting the k closest pairs of points that is based on the Lenhof et al. [15] algorithm.

Regarding the k closest/farthest bichromatic pairs problems, Salowe [18] proposed an algorithm with $O(n\log^2 n)$ running time that works only for rectilinear distance metric. Katoh and Iwano [14] gave a $O(\min(n^2, n\log n + k^{4/3}\log n/(\log k)^{1/3}))$ runtime solution which requires $O(n + k^{4/3}/(\log k)^{1/3} + k\log n)$ space. Both papers assumed that $m = n$.

In this paper we present fast approximation algorithms for the problems above. For a distance d determined by some pair of points in S and for any fixed $0 < \delta_1 \leq 1$, $\delta_2 \geq 1$, the distance d' is the (δ_1, δ_2)-*approximation* of d, if $\delta_1 d \leq d' \leq \delta_2 d$. We give an $O(n\log^3 n)$ runtime solution for the distance selection problem that computes a pair of points realizing distance d' that is either $(1, 1+\varepsilon)$ or $(1-\varepsilon, 1)$-approximation of the actual k-th distance, for any fixed $\varepsilon > 0$. We also present an $O(n\log n)$ time algorithm for computing the $(1-\varepsilon, 1+\varepsilon)$-approximation of k-th distance. It compares well to the paper by Agarwal et al. [1] that considers a similar problem, where one want to identify approximate "median" distance, that is, a pair of points $p, q \in S$ with the property that there exist absolute constants c_1 and c_2 such that $0 < c_1 < \frac{1}{2} < c_2 < 1$ and the rank of the distance determined by p and q is between $c_1\binom{n}{2}$ and $c_2\binom{n}{2}$. They [1] showed how to solve this problem in $O(n\log n)$ time.

We also show how to extend our solution in order to answer efficiently the queries approximating k-th distance. We provide an $O(n\log^3 n + k)$ runtime and $O(n\log n)$ space algorithm for reporting k distinct pairs of points whose distances $d'_1 \leq d'_2 \leq \ldots \leq d'_k$ are $(1-\varepsilon, 1)$-approximation of k largest distances, i.e.

$$(1-\varepsilon)d_{\binom{n}{2}+i-k} \leq d'_i \leq d_{\binom{n}{2}+i-k}.$$

We achieve the same runtime performance for reporting smallest k distances with $(1, 1+\varepsilon)$-approximation and propose algorithms for the reporting and enumerating bichromatic distances with the same running time.

We present an algorithm for the approximation version of the distance ranking problem. Let K be the solution of the distance ranking problem. For a given $\varepsilon > 0$, we find two numbers k_1, k_2, such that $0 \leq k_1 \leq K \leq k_2 \leq \binom{n}{2}$ and

$$1 - \varepsilon \leq d_{k_1} \leq 1 \leq d_{k_2} \leq 1 + \varepsilon.$$

The running time of our algorithm is $O(n \log^3 n)$ and it can be extended to deal with a *separator* query: instead of considering distances less than one, we may consider the distances less than a query separator s. A query time is $O(\log n)$. The distance ranking problem is closely related to the well-known repeated unit distances problem[17], in which one wants to determine the number of pairs of points at unit distance in the plane. The best known lower bound (supposed to be tight) is $n^{1+c/\log\log n}$ for some appropriate constant $c > 0$ and the best upper bound is $O(n^{4/3})$. We also show that the runtime and space requirements of our approximation algorithms for the problems above can be improved when dealing with L_∞ metric.

Our method can be extended to deal with more general problems, that we call *range distance selection* and *batched distance selection*. In the range distance selection we are given two numbers k_1, k_2, $1 \leq k_1 \leq k_2 \leq \binom{n}{2}$ and we want to report all the pairs of points whose distance rank is between k_1 and k_2. In the batched distance selection problem we are given an increasing sequence of numbers $1 \leq k_1 \leq k_2, \ldots, \leq k_l \leq \binom{n}{2}$, we want to report all the pairs of points realizing k_1-th, k_2-th, $\ldots k_l$-th distance. For our best knowledge nothing has been done regarding these two problems. We show the efficient $(1-\varepsilon, 1+\varepsilon)$-approximation algorithms for these problems.

The main contribution of this paper is by developing efficient approximating algorithms for the several well known-optimization problems using an unifying approach that based on well separated pairs decomposition introduced by Callahan and Kosaraju [6]. This paper is organized as follows. In the next section we briefly describe well-separated pair decomposition. Section 3 is dedicated to the distance selection problem, its query version and to the range and batched distance selection problems. We consider the rest of the distance problems in Section 4.

2 Well-separated pair decomposition

In this section we shortly describe the well-separated pair decomposition proposed by Callahan and Kosaraju [6]

Let A and B be two sets of points in d-dimensional space ($d \geq 1$) of size n and m, respectively. Let s be some constant strictly greater than 0 and let $R(A)$ (resp. $R(B)$) be the smallest axis-parallel bounding box that encloses all the points of A (resp. B). We say that point sets A and B are *well-separated* with respect to s, if $R(A)$ and $R(B)$ can be each contained in d-dimensional ball of some radius r, such that the distance between these two ball is at least sr. One can easily show that for a given two well-separated sets A and B, if $p_1 \in A$, $p_2, p_3 \in B$ then $dist(p_1, p_2) \leq (1 + \frac{2}{s}) dist(p_1, p_3)$. (For L_∞ metric the previous inequality looks $dist(p_1, p_2) \leq (1 + \frac{2\sqrt{2}}{s}) dist(p_1, p_3)$. For general L_p metric the inequality may differ by some multiplicative constant.)

Let S be a set of d-dimensional points, and let $s > 0$. A *well-separated pair decomposition* (WSPD) for S with respect to S is a set of pairs
$\{(A_1, B_1), (A_2, B_2), \ldots, (A_p, B_p)\}$ such that:

(i) $A_i \subseteq A$ and $B_i \subseteq B$, for all $i = 1, \ldots, p$.

(ii) $A_i \cap B_i = \emptyset$, for all $i = 1, \ldots, p$.

(iii) A_i and B_i are well-separated with respect to s.

(iv) each unordered pair of points $(p, q), p, q \in S, p \neq q$, there is exactly one pair (A_i, B_i) in a set $\{(A_1, B_1), (A_2, B_2), \ldots, (A_p, B_p)\}$ such that either $p \in A_i$ and $q \in B_i$ or $p \in B_i$ and $q \in A_i$.

The main idea of the algorithm for construction WSPD is to build a binary split tree T whose leaves are points of S, with internal nodes corresponding to subsets of S. Each pair (A_i, B_i) in WSPD is represented by two nodes $v, u \in T$, such that all the leaves in the subtree rooted at by v correspond to the points of A_i and all the leaves in the subtree rooted at by u correspond to the points of B_i.

The paper of Callahan and Kosaraju [6] presents an algorithm that implicitly constructs WSPD for a given set S and separation value $s > 0$ in $O(n \log n + s^2 n)$ time such that the number of pairs (A_i, B_i) is $O(s^2 n)$. Moreover, Callahan [5] showed to compute $WSPD$ in which at least one of the sets A_i, B_i of each pair (A_i, B_i) contains exactly one point of S. The running time remains the same; however, the number of pairs increases to $O(n \log n + s^2 n)$.

3 Approximating k-th distance

We first describe a general scheme for obtaining $(1, 1+\varepsilon)$ or $(1-\varepsilon, 1)$-approximation of k-th distance and then show how to get an algorithm for the query version of the problem. At the end we obtain a simpler algorithm for the $(1-\varepsilon, 1+\varepsilon)$-approximation and show how to extend it for the batched (range) selection problem.

3.1 General approximation scheme

Our algorithm consists of several stages. At the first stage we compute a WSPD for S with separation constant $s = \frac{2}{\varepsilon}$. As it was mentioned in the Section 2 each pair (A_i, B_i) in WSPD is represented by two nodes in T and $|A_i| = 1$ or/and $|B_i| = 1$. Next, for each pair (A_i, B_i), $1 \leq i \leq p, p = O(n \log n)$ we compute the following values:

- $m_i = \min_{a \in A_i, b \in B_i} dist(a, b)$.
- $M_i = \max_{a \in A_i, b \in B_i} dist(a, b)$.
- $(a_i, b_i) \in (A_i, B_i)$ that realizes M_i, i.e. $dist(a_i, b_i) = M_i$.
- $\alpha_i = |A_i||B_i|$.

In other words, $m_i(M_i)$ is the minimal (maximal) distance between points that belong to A_i and B_i. The value α_i is the total number of distinct pairs (a, b), $a \in A_i$, $b \in B_i$.

We sort all m_i, $1 \leq i \leq p$ in increasing order. Without loss of generality, we assume that $m_1 \leq m_2 \leq \ldots \leq m_p$. Out task now is to find the smallest index j, such that $\Sigma_{i=1}^{j} \alpha_i \geq k$. We claim that $M' = \max_{i=1}^{j} M_i$ is the $(1, 1+\varepsilon)$-approximation of k-th distance. In what follows we prove the correctness of our algorithm and show how to implement it efficiently.

Lemma 1 *The value M' is the $(1, 1+\varepsilon)$-approximation of k-th distance.*

Proof. We need to show that $d_k \leq M' \leq (1+\varepsilon) d_k$. In order to show the left inequality we observe that the total number of distances defined by pairs (A_i, B_i), $1 \leq i \leq j$ is at least k because $\Sigma_{i=1}^{j} \alpha_i \geq k$. Since M' is the maximum of these distances $M' \geq d_k$ follows.

We recall that all possible pairs of points of S are uniquely represented by pairs (A_i, B_i) in WSPD. Consider the set of pairs $D = \{(a,b)|a \in A_i, b \in B_i, i \geq j\}$. By definition of j, m_j is the smallest distance defined by pairs of D. The total number of pairs in D is at least $\binom{n}{2} - k$. Thus, the pair of points (p,q) that defines d_k belongs to D. Therefore, $d_k \geq m_j$. Let t, $1 \leq t \leq j$ be the index such that $M' = M_t$. ¿From the observation in previous section it follows that $M_t \leq (1 + \frac{2}{s})m_t = (1+\varepsilon)m_t$. Thus,

$$M' \leq (1+\varepsilon)m_t \leq (1+\varepsilon)m_j \leq (1+\varepsilon)d_k$$

∎

It remains to show how to implement this algorithm efficiently, i.e. how to compute the values m_i, M_i, α_i, $1 \leq i \leq p$. First we show how to compute α_i. In other words we need to compute the cardinalities of A_i and B_i, $1 \leq i \leq p$. Recall that each pair (A_i, B_i) in WSPD is represented by two nodes v_i, u_i of the split tree T. The cardinality of $A_i(B_i)$ equals to the number of leaves in the subtree of T rooted at $v_i(u_i)$. Thus, by postorder traversal of T we are able to compute all the required cardinalities.

Next, we describe how to compute the values m_i, $1 \leq i \leq p$ (the similar algorithm works for M_i as well). Without loss of generality, we assume the singleton set of each pair (A_i, B_i) in WSPD is $A_i = \{a_i\}$. Our problem, thus, for each a_i, $1 \leq i \leq p$ to compute the nearest neighbor in corresponding B_i. The naive approach would lead to an $O(n^2)$ algorithm since the total complexity of all B_i's is $O(n^2)$. In order to find the neighbor for each a_i we use Voronoi Diagrams for B_i with corresponding point location data structures. We cannot compute explicitly all Voronoi Diagrams because of total complexity of all B_i. Fortunately, we can maintain dynamically Voronoi Diagrams while traversing a split tree T in a bottom-up fashion. We proceed as follows. For each node v of the split tree T that corresponds to we create a associate list L_v containing all a_i's such that $(\{a_i\}, B_i)$ belongs to WSPD and B_i is represented by v. Note that some lists contain at least $O(\log n)$ elements since the total number of pairs in WSPD is $O(n \log n)$ but the number of nodes in T is linear. We traverse a tree T in a postorder fashion starting from leaves. At any time we have a *front* that is a set of not processed yet nodes such that all the nodes below them are processed already. Recall that S_v is a subset of S associated with a node v in T. The maintenance of the Voronoi Diagram of S_v leads to quadratic runtime for biased trees. Instead we use a partition R_v of S_v into disjoint sets $S_v^1, \ldots, S_v^q$ and maintain the Voronoi Diagram VD with corresponding point location data structure PL for each set S_j, $1 \leq j \leq q$ in R_v. The sizes of the sets in R_v are different and restricted to be the powers of two. As the consequence the number of such sets is at most $\log n$, i.e. $q \leq \log n$.

We describe how to process a node v in T. In v is a leaf we build a corresponding Voronoi Diagram VD with a structure PL. To process the internal node we use the Voronoi Diagrams with a associated PL structures of its sons. In order to obtain a partition S_v we merge data structures VD and PL of equal sizes. It can be done, for example, by simple rebuilding a new data structures for the merged set of points. After merging all data structures, for each $a_i \in L_v$, we find a nearest neighbor in S_v using the point location data structures of v.

Considering time and space complexity. All steps in our algorithm excluding computing m_i, M_i, $1 \leq i \leq p$ can be carried out in $O(n \log n)$ time using $O(n \log n)$ space. The maintenance of the Voronoi Diagrams with a point location data structures takes $O(n \log^2 n)$ time since only $\log n$ merges are possible. Determining the nearest neighbor for each a_i, $1 \leq i \leq p$ consumes in total $O(n \log^3 n)$ time because $p = O(n \log n)$, S_v is represented by $\log n$ data structures and an additional $\log n$ comes from the point location query (the same holds for computing farthest neighbor for each

a_i, $1 \leq i \leq p$). The space requirements remain $O(n \log n)$ since one can easily observe that each point of S appears in front exactly one, providing only $O(n)$ additional space. Notice that at the same time and space we can find a pair of points in S that realizes $(1, 1+\varepsilon)$-approximation of d_k.

Theorem 2 *Given a set S be a set of n points in the plane, number k, $1 \leq k \leq \frac{n(n-1)}{2}$ and $\varepsilon > 0$ in time $O(n \log^3 n)$ we can compute a pair of point realizing distance d' that is an $(1, 1+\varepsilon)$-approximation of the actual k-th distance.*

Remark. The same scheme works for $(1-\varepsilon, 1)$-approximation. The difference is that instead of considering m_i, $1 \leq i \leq p$, we sort M_i. The index j is defined to be the largest index such that $\Sigma_{i=j}^{p} \alpha_i \geq \binom{n}{2} - k$. Choose the $m' = \min_{i=j}^{p} m_i$ as $(1-\varepsilon, 1)$-approximation of d_k.

3.2 Distance by query

Using the approximation algorithm above we show how to solve distance by query problem. Indeed, we can build a binary tree T_α with the leaves corresponding to $\alpha_1, \ldots, \alpha_p$. Each internal node $v \in T_\alpha$ will keep three values: $\Sigma_{i=l_1}^{l_2} \alpha_i$, $\Sigma_{i=l_2+1}^{l_3} \alpha_i$, where $\alpha_{l_1}, \ldots, \alpha_{l_2}$ $(\alpha_{l_2+1}, \ldots, \alpha_{l_3})$ are the values that correspond to the leaves of the left subtree (resp. right subtree) of a tree rooted by v, and the third value $M_v = \max_{i=l_1}^{l_3} M_i$. Clearly, the construction of this tree T_α with the augmented values can be computed in linear time. We associate with each node $v \in T_\alpha$ an index j_v, such that α_{j_v} corresponds to the rightmost leaf in the subtree rooted at v. Given a value k, we traverse T_α starting from the root towards its children. We need to find a node u, with the smallest j_u such that $\Sigma_{i=1}^{j_u} \alpha_i \geq k$. It can be done in $O(\log n)$ time, by simple keeping the total number of nodes to the left of the current searching path. At each node where the path goes right, we collect the value M_v stored in the left subtree. At the end, we report the maximal of the collected M_v values.

Lemma 3 *Given $\varepsilon > 0$, WSPD can be preprocessed in $O(n \log^3 n)$ time, such that given a number k we can answer two types of approximate distance queries ($(1-\varepsilon, 1)$, $(1, 1+\varepsilon)$) in $O(\log n)$ time.*

3.3 $(1-\varepsilon, 1+\varepsilon)$-approximation

We can simplify our algorithm for $(1, 1+\varepsilon)$-approximation of the k-th distance. Using WSPD for S with separation constant $s = \frac{2}{\varepsilon}$ we compute only α_i and take any pair $(a_i, b_i) \in (A_i, B_i), 1 \leq i \leq p$, $p = O(n)$. The $(1-\varepsilon, 1+\varepsilon)$-approximations of k-th distances for all $1 \leq k \leq \binom{n}{2}$ can be taken among p pairs (a_i, b_i). To show it first we sort the distances d'_i between a_i and b_i, $1 \leq i \leq p$. We assume that the pairs (A_i, B_i) are in order of increasing d'_i. Let M_l be the maximal distance defined by pairs of points in (A_l, B_l). For a particular k we compute the smallest j such that $\sum_{i=1}^{j} \alpha_i \geq k$. The points a_j and b_j are at distance at most $M' = \max_{1 \leq i \leq j} M_i$. Similarly to Lemma 1 we have

$$d'_j \leq M' \leq (1+\varepsilon) d_k. \tag{1}$$

From other hand,

$$(1+\varepsilon) d'_j = \max_{1 \leq i \leq j} (1+\varepsilon) d'_i \geq \max_{1 \leq i \leq j} M_i = M' \geq d_k. \tag{2}$$

Thus, $d'_j \geq (1-\varepsilon) d_k$ and (a_j, b_j) is $(1-\varepsilon, 1+\varepsilon)$-approximation of the k-th distance.

Theorem 4 *The $(1-\varepsilon, 1+\varepsilon)$-approximation of the k-th distance among a given n points in $R^d, d \geq 1$ can be computed in $O(n \log n)$ time and $O(n)$ space. Moreover, we can answer $(1-\varepsilon, 1+\varepsilon))$-approximate distance queries in $O(\log n)$ time in any $d \geq 1$. The preprocessing time is $O(n \log n)$.*

Proof. We sort the linear number of distances d'_i, $1 \leq i \leq p$ in $O(n \log n)$ time. We perform a binary search over the sorted list of values. In $O(1)$ time we can compute the median distance $d'_m, 1 \leq m \leq p$. We partition the set of indices $I = \{1 \leq i \leq p\}$ into $C_{\leq} = \{i | d'_i \leq d'_m\}$ and $C_{>} = I \setminus C_{\leq}$. Compute $\alpha' = \sum_{i \in C_{\leq}} \alpha_i$. If $\alpha' \leq k$ then we replace all pairs $(a_i, b_i), i \in C_{\leq}$ by a pair with largest distance and assign α' for it. If $\alpha' < k$ then we replace all pairs $(a_i, b_i), i \in C_{>}$ by a pair with the smallest distance and assign α' for it. It reduces the number candidate pairs by half. The total time to find the $(1 - \varepsilon, 1 + \varepsilon)$-approximation of the k-th distance (after sorting d'_i distances) is $O(\log n)$ using. a tree T_α as in Section 3.2 for efficient computation sums of α_i. ∎

Remark. From the previous theorem it follows that we can approximate the *diameter* of the points (i.e. the largest pairwise distance between points) in $O(n \log n)$ time as well as the smallest pairwise distance between the points.

3.4 Range and batched distance selection

We note that these two problems can not be solved using our approximation distance selection query algorithm, since it can report the same pair of points few times. As a matter of fact the batched distance selection problem is more general than the range selection (except for the length of input). We explain an algorithm for the batched version of the problem that works as well for the range version. We proceed as in Section 3.3, by taking arbitrary pairs of points (a_i, b_i) from $(A_, B_i)$, $1 \leq i \leq p$ of WSPD and sorting them according to the distances. We find smallest j such that $K = \sum_{i=1}^{j} \alpha_i \geq k_1$. We find the number t_1 of all the indices in the sequence $k_1, k_2, \ldots, k_l$ that are less or equal to K. We take any t_1 pairs of points from (A_j, B_j) as a part of our solution approximating k_1-th, …, k_{t_1}-th distances. For the remaining indices k_{t_1+1}, …, k_l we apply the same scheme. The correctness of this algorithm follows from the discussion in Section 3.3.

Theorem 5 *The $(1 - \varepsilon, 1 + \varepsilon))$-approximation of the batched (range) distance selection problem with points in $R^d, d \geq 1$ can be found in $O(n \log n)$ time and $O(n)$ space.*

4 Other distance problems

In this section we show how to solve the rest of the distance problems described at Introduction. We first present algorithm for reporting k distances (smallest or largest) and then extend it for the bichromatic case. Next we show how to solve the distance ranking problem and its query version. Finally, we observe how we can speed up our algorithms for L_∞ metric and generalize them for d-dimensional space.

4.1 Reporting k largest distances

Notice that we cannot apply k times k-th distance selection query algorithms since we are interested in the solution with k distinct pair of points. We compute WSPD for S with separation constant $s = \frac{2}{\varepsilon}$. Let m_i, M_i and α_i, $1 \leq i \leq p$ be as described in Section 3. We sort all M_i, $1 \leq i \leq p$ in increasing order. Let us assume that $M_1 \leq M_2 \leq \ldots \leq M_p$. We find the largest index j such that $\Sigma_{i=j}^{p} \alpha_i \geq k$. Suppose that $\Sigma_{i=j+1}^{p} \alpha_i = k'$ $(k' < k)$. Denote by A set of $k - k'$ any distinct pairs of points defined by (A_j, B_j). Next, we show that the set R of all distinct pairs points defined by $(A_{j+1}, B_{j+1}), \ldots, (A_p, B_p)$ and by A is a required solution. Assume, that that we all the distances defined by pairs of points in R are sorted, i.e. $d'_1 \leq d'_2 \leq \ldots \leq d'_k$.

Lemma 6 $(1-\varepsilon)d_{\binom{n}{2}+i-k} \leq d'_i \leq d_{\binom{n}{2}+i-k}, 1 \leq i \leq k$.

Proof. Fix i. Since there are at least $k-i$ distances $d'_{i+1}, \ldots, d'_k$ that are larger than d'_i, we conclude that $d'_i \leq d_{\binom{n}{2}+i-k}$. From the other hand, assume that d'_i is realized by some pair of points defined by (A_l, B_l), for some $1 \leq l \leq p$. Obviously, $M_l \leq (1+\varepsilon)d'_i$ by observation from Section 2. However, as one can see the value M_l is larger (or equal) than at least $\binom{n}{2}+i-k$ different distances. Hence, $M_l \geq d_{\binom{n}{2}+i-k}$ and we have $d_{\binom{n}{2}+i-k} \leq (1+\varepsilon)d'_i$. It remains to point out that $\frac{1}{1+\varepsilon} > 1-\varepsilon$ and, therefore, $(1-\varepsilon)d_{\binom{n}{2}+i-k} \leq d'_i$. ∎

In order to implement the above algorithm efficiently we use the same data structures as we used for k-th distance selection algorithm. The rest is straightforward.

Theorem 7 *Given a set S be a set of n points in the plane, number k, $1 \leq k \leq \frac{n(n-1)}{2}$ and $\varepsilon > 0$ in time $O(n \log^3 n + k)$ we can find k distinct pairs of points whose distances are $(1-\varepsilon, 1)$-approximation of the actual k largest distances.*

Remark. The same scheme works for the reporting $(1, 1+\varepsilon)$-approximate k smallest distances.

4.2 Reporting bichromatic distances

Basically, the idea is similar to the one used in Section 4.1. We consider a set $S = A \cup B$ and perform almost the same operations as before. We compute a WSPD for S with $s = \frac{2}{\varepsilon}$. As before, without loss of generality we assume that for each pair (A_i, B_i), $1 \leq i \leq p$, $|A_i| = 1$. Thus, A_i may contain exactly one red or exactly one blue point. Next, for each pair (A_i, B_i) we compute the values m_i, M_i and α_i, but now considering only bichromatic distances, that is, the distances that defined by a point from A_i and by points with different color from B_i. We proceed as in Section 4.1. We only need to explain how to compute the values m_i, M_i and α_i, $1 \leq i \leq p$. By a traversal of a tree T which represents a WSPD we can count for each node $v \in T$ the number of blue and red leaves in the subtree rooted at v. It will define our α_i, $1 \leq i \leq p$. In order to deal with m_i, M_i, instead of maintaining two Voronoi Diagrams for each node $v \in T$: closest and farthest neighbor, we will maintain four Voronoi Diagrams: closest and farthest neighbor for each color of points. The rest follows immediately.

Theorem 8 *Given two sets of points R and B in the plane, with cardinalities n and m, respectively, a number k, $1 \leq k \leq \frac{n(n-1)}{2}$ and $\varepsilon > 0$ in time $O(\max(m,n) \log^3 \max(m,n) + k)$ we can find k distinct pairs of red-blue points whose bichromatic distances are $(1-\varepsilon, 1)$-approximation ($(1, 1+\varepsilon)$-approximation) of the actual k largest (smallest) bichromatic distances.*

4.3 Approximate distance ranking

The approach to approximate the rank of unit distance is similar to approximating the k-th distance. We again use a WSPD for S with separation constant $s = \frac{2}{\varepsilon}$. For each pair $\{A_i, B_i\}$, $1 \leq i \leq p$ we compute $m_i, M_i, (a_i, b_i)$, and α_i as in Section 3. The set of indices $I = \{1 \leq i \leq p\}$ is partitioned into 3 subsets $C_< = \{i \in I | M_i < 1\}$, $C_> = \{i \in I | m_i > 1\}$, and $C_= = I \setminus C_< \cup C_>$. If $C_=$ is empty then the distances formed by the pairs $(A_i, B_i), i \in C_<$ are closer than 1 and the remaining distances are larger than 1. Hence the number of pairs of points in S at distance less than 1 is $\sum_{i \in C_<} \alpha_i$. Otherwise we set $k_1 = 1 + \sum_{i \in C_<} \alpha_i$ and $k_2 = \binom{n}{2} - \sum_{i \in C_>} \alpha_i$. The numbers k_1 and k_2 can be computed in linear time.

Lemma 9 *The numbers k_1 and k_2 approximate the rank of unit distance.*

Proof. It is clear that $0 \leq k_1 \leq K \leq k_2 \leq \binom{n}{2}$, where K is the exact solution. The definition of k_1 and k_2 implies $d_{k_1} \leq 1 \leq d_{k_2}$. It remains to show the k_1-th and k_2-th distances are close to 1. Let $(a_{i_1}, b_{i_1}) \in (A_{i_1}, B_{i_1})$ and $(a_{i_2}, b_{i_2}) \in (A_{i_2}, B_{i_2})$ be the pairs of points defined these distances. Note that $i_1, i_2 \in C_=$ by choice of k_1 and k_2. Recall that, for any i, the sets A_i and B_i are well-separated with the separation constant $s = \frac{2}{\varepsilon}$. Hence $d_{k_2} \leq M_{i_2} \leq (1+\varepsilon)m_{i_2} \leq 1+\varepsilon$ and $d_{k_1} \geq m_{i_1} \geq \frac{M_{i_1}}{(1+\varepsilon)} \geq \frac{M_{i_1}}{(1+\varepsilon)} \geq 1-\varepsilon$. The lemma follows. ∎

4.3.1 Approximate rank query

In the rank query, for a separator s, we find two numbers k_1, k_2, such that $0 \leq k_1 \leq K \leq k_2 \leq \binom{n}{2}$ and $(1-\varepsilon)s \leq d_{k_1} \leq s \leq d_{k_2} \leq (1+\varepsilon)s$ where K is exact rank of the distance s. Notice that we can apply a result from Section 3 for a distance by query problem and obtain an $O(\log^2 n)$ query time for this problem. We show how to do better. Let σ^1 be the permutation of M_i values, such that $M_{\sigma^1_1} \leq M_{\sigma^1_2} \leq \ldots \leq M_{\sigma^1_p}$. Similarly, σ^2 is the permutation for the sorted m_i values. The possible approximate lower ranks are stored in the array $MRANK$. The i-th element of $MRANK$ is $\Sigma_{j=1}^{i} \alpha_{\sigma^1_j}$. The array $MRANK$ can be computed in $O(p) = O(n \log n)$ time. We also use an array $mRANK$, whose i-th element is $\Sigma_{j=i+1}^{p} \alpha_{\sigma^2_j}$. The approximate rank query can be answered as follows. We locate the separator s among the sorted $M_i, 1 \leq i \leq p$ values and find the largest index l such that $M_{\sigma^1_l} < s$. It can be done in $O(\log p) = O(\log n)$ time. We also find the smallest index t with $m_{\sigma^2_t} > s$. The condition $C_= = \emptyset$ implies $MRANK[l] + mRANK[t] = \binom{n}{2}$ and the exact solution is equal to $MRANK[l]$ in this case. Otherwise, set $k_1 = 1 + MRANK[l]$ and $k_2 = mRANK[t]$.

Lemma 10 *The WSPD can be preprocessed in $O(n \log^3 n)$ time to answer approximate rank queries in $O(\log n)$ time.*

4.4 L_∞ case

We first consider the problems of computing the $(1, 1+\varepsilon)$-approximation of distances under L_∞ metric. We take $s = \frac{2\sqrt{2}}{\varepsilon}$. Notice, that we avoid the computation of $m_i, 1 \leq i \leq p$ values by taking arbitrary pairs of points (a_i, b_i) from each $(A_i, B_i), 1 \leq i \leq p$ (see equation 1). As in Section 3.3 we sort these pairs of points by distances (we assume that index i of (a_i, b_i) corresponds to the rank of $dist(a_i, b_i)$ in the sorted order). The value M' computed in Section 3.3 is the $(1, 1+\varepsilon)$-approximation of k-th distance. Still, we need to compute M_i values, $1 \leq i \leq p$. In d-dimensional space ($d \geq 1$) under L_∞ the values M_i can be computed efficiently without using Voronoi Diagrams. The points defining M_i should lye on the boundary of the smallest axis-parallel bounding box of set $A_i \cup B_i$. Recall that A_i and B_i are well separated and, thus, the L_∞ diameter of $A_i \cup B_i$ is defined by a pair (p, q) such that $p \in A_i$ and $q \in B_i$. Instead of maintaining Voronoi Diagram in bottom-up traversal of tree T we maintain the bounding boxes for sets of points corresponding to the nodes of T. The new bounding box can be computed in $O(1)$ time using the information from the previous steps. Moreover, we can use a WSPD with $p = O(n)$.

Thus, we conclude by

Theorem 11 *The running time of $(1, 1+\varepsilon)$-approximation scheme applied to distance selection, reporting distances (monochromatic or bichromatic), distance ranking problems under L_∞ metric in* *any d-dimensional space ($d \geq 1$) is improved to $O(n \log n)$, $O(n \log n + k)$, $O(n \log n)$, respectively, using only linear space.*

Regarding $(1-\varepsilon, 1)$ algorithms we notice that the computation of M_i can be avoided using the equation 2. As above we sort $dist(a_i, b_i), 1 \leq i \leq p$. The index j is defined to be the largest index such that $\Sigma_{i=j}^{p} \alpha_i \geq \binom{n}{2} - k$. Similarly, to the $(1, 1+\varepsilon)$ case we choose $m' = \min_{i=j}^{p} m_i$ as $(1-\varepsilon, 1)$-approximation of d_k.

The computation of $m_i, 1 \leq i \leq p$ can be done similarly to the approach described in [4]. We use a WSPD with $p = O(n)$ and assume $A_i = \{a_i\}$, $1 \leq i \leq p$. For each point a_i we need to find the closest neighbor in corresponding B_i. Let l_1 be a line whose slope is 45° passing through the a_i and l_2 a a line whose slope is 135° passing through the a_i. These lines define four wedges: $Q_{top}, Q_{bottom}, Q_{left}, Q_{right}$. For any point p lying in $Q_{left} \cup Q_{right}(Q_{bottom} \cup Q_{top})$ the L_∞-distance to a_i is defined by the x-distance (y-distance, resp.) to a_i. We perform four range queries, using orthogonal range tree [3] data structure (in coordinate system defined by lines l_1, l_2), each of them corresponding to the appropriate wedge. For each node in a secondary data structure we keep four values $x_{min}, x_{max}, y_{min}, y_{max}$ (computed in the initial coordinate system) of points in corresponding range. Consider for a example wedge Q_{right}. Our query corresponding to Q_{right} marks $O(\log^2 n)$ nodes. The minimum of x_min values stored in these nodes define the closest neighbor point to a_i lying in Q_{right}. We proceed similarly with the other wedges. We maintain orthogonal range tree data structures dynamically in a bottom-up fashion while traversing split tree T. In order to merge two data structures we simply insert all the points stored in the smaller range tree into the larger one. Notice, that each point can be inserted at most $O(\log n)$ time. Each insertion takes $O(\log^2 n)$ time. The total time for maintaining the range trees and computing m_i, $1 \leq i \leq p$ is $O(n \log^3 n)$. It can generalized to d-dimensional space, $d > 2$ (in contrast to other metrics).

Theorem 12 *The running time of $(1-\varepsilon, 1)$-approximation scheme applied to distance selection, reporting distances (monochromatic or bichromatic), distance ranking problems under L_∞ metric in any d, $d \geq 1$ dimensional space is $O(n \log^{d+1} n)$, $O(n \log^{d+1} n + k)$, $O(n \log^{d+1} n)$, respectively.*

Remark. The running times in the theorem above can be improved slightly by $O(\frac{\log n}{\log \log n})$ factor using dynamic fractional cascading technique [16].

References

[1] P. Agarwal, B. Aronov, M. Sharir, S. Suri, "Selecting distances in the plane", *Algorithmica*, 9, pp. 495–514, 1993.

[2] P. Agarwal, M. Sharir, E. Welzl "The discrete 2-center problem", *Proc. 13th ACM Symp. on Computational Geometry*, pp. 147–155, 1997.

[3] M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf "Computational Geometry: Algorithms and Applications", Springer-Verlag, 1997.

[4] S. Bespamyatnikh, K. Kedem, M. Segal "Optimal Facility Location under Various Distance Function", in *Workshop on Algorithms and Data Structures'99*, pp. 318–329, 1999.

[5] P. Calahan "Dealing with higher dimensions: the well-separated pair decomposition and its applications", Ph.D thesis, Johns Hopkins University, USA, 1995.

[6] P. Callahan and R. Kosaraju "A decomposition of multidimensional point sets with applications to k-nearest neighbors and n-body potential fields", *Journal of ACM*, 42(1), pp. 67–90, 1995.

[7] T. Chan "On enumerating and selecting distances", In *Proc. 14th Annu. ACM Sympos. Comput. Geom.*, pp. 279–286, 1998.

[8] M. Dickerson and D. Eppstein "Algorithms for proximity problems in higher dimensions", *Computational Geometry: Theory and Applications* 5, pp. 277–291, 1996.

[9] M. Dickerson, R. L. Scot Drysdale, J-R. Sack "Simple algorithms for enumerating interpoint distances and finding k nearest neighbors", *Int. J. Comput. Geom. and Appls.*, 2(3), pp. 221–239, 1992.

[10] M. Dickerson and J. Shugart "A simple algorithm for enumerating longest distances in the plane", *Inf. Process. Lett.* 45, pp. 269–274, 1993.

[11] J. Erickson, "On the Relative Complexities of Some Geometric Problems", in *Proc. 7th Canad. Conf. Comput. Geom.*, pp. 85–90, 1995.

[12] M. Goodrich, "Geometric partitioning made easier, even in parallel", *Proc. 9th Annu. ACM Sympos. Comput. Geom.*, pp. 73-82, 1993.

[13] M. Katz and M. Sharir "An expander-based approach to geometric optimization", *SIAM J. Comput.*, 26(5), pp. 1384–1408, 1997.

[14] N. Katoh, K. Iwano, "Finding k farthest pairs and k closest/farthest bichromatic pairs for poin ts in the plane", *Int. J. Comput. Geom. Appl.* 5, pp. 37–52, 1995.

[15] H-P. Lenhof and M.Smid "Sequential and parallel algorithms for the k closest pairs problem", *Internat. J. Comput. Geom. Appls.* 5, pp. 273–288, 1995.

[16] K. Mehlhorn and S. Näher, "Dynamic fractional cascading", *Algorithmica*, 5, pp. 215–241, 1990.

[17] J. Pach and P. Agarwal "Combinatorial Geometry", John Wiley & Sons, Inc, 1995.

[18] J. Salowe, "L_∞ interdistance selection by parametric searching", *Inf. Process. Lett.*, 30, pp. 9–14, 1989.

[19] J. Salowe, "Enumerating interdistances in space", *Int. J. Comput. Geom. Appls.*, 2, pp. 49–59, 1992.

[20] M. Segal, K. Kedem, "Geometric applications of posets", *Computational Geometry: Theory and Applications*, 11, pp. 143–156, 1998.