On Enumerating and Selecting Distances*

Timothy M. Chan†

Department of Mathematics and Computer Science
University of Miami, Coral Gables, FL 33124–4250, USA

June 17, 1999

Abstract

Given an n-point set, the problems of enumerating the k closest pairs and selecting the k-th smallest distance are revisited. For the enumeration problem, we give simpler randomized and deterministic algorithms with $O(n \log n + k)$ running time in any fixed-dimensional Euclidean space. For the selection problem, we give a randomized algorithm with running time $O(n \log n + n^{2/3} k^{1/3} \log^{5/3} n)$. We also describe output-sensitive results for halfspace range counting that are of use in more general distance selection problems. None of our algorithms requires parametric search.

Keywords: distance enumeration, distance selection, closest pairs, range counting, randomized algorithms.

1 Introduction

Finding the *closest pair* of an n-point set has a long history in computational geometry (see [34] for a nice survey). In the plane, the problem can be solved in $O(n \log n)$ time using the Delaunay triangulation. In an arbitrary fixed dimension d, the first $O(n \log n)$ algorithm, based on divide-and-conquer, was described by Bentley and Shamos [5]. Another $O(n \log n)$ algorithm of Vaidya [35] can actually find the nearest neighbor to each of the given points. In 1976, Rabin [32] suggested a random sampling method that requires only $O(n)$ expected time assuming a RAM model of computation that supports the floor function and constant-time hashing (see also [15, 18, 23]). Under the algebraic decision tree model, linear randomized complexity is still attainable when the points have been pre-sorted along each of the d coordinates.

The more general problem of *enumerating the k closest pairs* (or enumerating the first k smallest distances) has also received much attention. One of the earliest reported algorithms is by Dickerson *et al.* [12], who used the Delaunay triangulation to enumerate the k closest pairs in $O((n+k) \log n)$ time in two dimensions. (Their algorithm actually enumerates the distances in sorted order.) Salowe [33] was the first to give an $O(n \log n + k)$ algorithm for any fixed dimension. His algorithm employs the

*A preliminary version of this work appeared in *Proc. 14th ACM Sympos. Comput. Geom.*, pages 279–286, 1998.

†New address: Department of Computer Science, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada, e-mail: `tmchan@math.uwaterloo.ca`

parametric searching technique [29] and Vaidya's all-nearest-neighbor method, making implementation difficult. Lenhof and Smid [24] subsequently pointed out a much simplified algorithm—the only geometric structure needed is a grid. Alternatives that apply more advanced geometric structures have been proposed by Dickerson and Eppstein [13] (higher-dimensional Delaunay triangulations) and Arya and Smid [4] (spanners).

In this paper, we add two more distance enumeration algorithms to the list:

- A further simplification of Lenhof and Smid's algorithm, running in $O(n \log n + k)$ time. In their algorithm, parametric search is avoided by using instead an implicit binary search on a matrix with sorted rows and columns. This binary search is carried out in the style of Frederickson and Johnson [20] and requires repeated calls to a linear-time subroutine for weighted medians. We replace these median computations with a simpler "approximate-and-refine" scheme. A description of the new algorithm can be found in the proof of Theorem 3.5.

- An even simpler algorithm based on random sampling. This can be regarded as an extension of Rabin's closest pair algorithm. The expected running time is $O(n \log n + k)$ and can be brought down to $O(n + k)$ for pre-sorted point sets (Theorem 3.3). The approach follows from a general result from Section 2.1, which also has an application to enumerating the k farthest pairs (see Section 4).

Our algorithms work under the algebraic decision tree model and are thus optimal in this model. Distances are not enumerated in sorted order.

A related problem is how to *select the k-th smallest distance*. In the plane (under the Euclidean metric), Agarwal *et al.* [1] solved a decision version of the problem in $O(n^{4/3} \log^{2/3} n)$ expected time. Applying parametric search, they then showed that the k-th smallest distance can be selected in $O(n^{4/3} \log^{8/3} n)$ expected time in dimension 2. Deterministic variants were considered by Goodrich [19] and Katz and Sharir [22]. Improvements in the logarithmic factors were noted briefly by Matoušek [25]; more importantly, he showed how parametric search can be avoided to obtain simpler algorithms that are randomized.

Our contributions to the distance selection problem can be summarized as follows:

- A randomized reduction to the decision problem without parametric search. Matoušek only described in detail his randomized technique for the slope selection problem. We state and prove the result under a general setting in Section 2.2.

- A way to obtain running time sensitive to k. Agarwal *et al.*'s bound is worse than the $O(n \log n + k)$ bound for the enumeration problem when $k \ll n^{4/3}$. We show that a hybrid approach in the planar case yields an expected running time of $O(n \log n + n^{2/3} k^{1/3} \log^{5/3} n)$, improving both results for most values of k between 1 and $\Theta(n^2)$. Any improvement to the k-insensitive bound automatically implies an improvement to the k-sensitive bound (see Theorem 3.6).

In passing, we note that de Berg and Schwarzkopf [6] and Pellegrini [31] have obtained similar bounds—near $O(n + n^{2/3} k^{1/3})$—for the problem of counting the k intersecting pairs of n given line segments. It is plausible that their techniques could lead to the same result for distance selection. However, our approach exploits special properties of distances and thus avoids "intersection-sensitive cuttings;" furthermore, our approach extends to higher dimensions.

Finally, Section 4 mentions connection of more general distance selection problems to *halfspace range counting*: how to preprocess a given n-point set so that one can quickly count the number k of points inside a given query halfspace. The problem has been studied by a number of researchers; see [2, 28] for surveys. In the planar case, $O(n^{1/2} \operatorname{polylog} n)$ query time is attainable after $O(n \log n)$ preprocessing time [26]. On the other hand, results on halfspace range reporting imply an output-sensitive query time of $O(\log n + k)$ with the same preprocessing time [9]. We point out that combining previous approaches properly yields a query time bound of $O(k^{1/2} n^{\varepsilon})$ for any constant $\varepsilon > 0$ in this planar case. The precise output-sensitive bounds in higher dimensions, including preprocessing/query-time tradeoffs, are given by Theorem 4.1. Although the proof of these output-sensitive bounds requires no new ideas (the main ingredients being Matoušek's "shallow cutting lemma" and "partition theorem for shallow hyperplanes" [27]), they are worth noting due to the fundamental nature of range searching.

2 General Randomized Reductions

In this section, we study the distance enumeration and selection problems in an abstract setting and describe general reductions that will be of use in some of the concrete algorithms of the next section.

Let U represent the *object space* and $d : U \times U \to \mathbb{R}$ represent the *distance function*, where $d(p, q) = d(q, p)$ for any $p, q \in U$.

Notation. Given sets $P, Q \subset U$, define the multiset $d(P, Q) = \{d(p, q) : (p, q) \in P \times Q\}$. For any integer $1 \le k \le |P| \cdot |Q|$, let $R(P, Q, k)$ be the k-th smallest element in $d(P, Q)$. For any $r \in \mathbb{R}$, let $K(P, Q, r) = |d(P, Q) \cap (-\infty, r]|$. Observe that $R(P, Q, k) \le r$ iff $K(P, Q, r) \ge k$. For notational convenience, set $d(P) = d(P, P)$, $R(P, k) = R(P, P, k)$, and $K(P, r) = K(P, P, r)$.

Given as input a set $P \subset U$ of size n, an integer k, and a number $r \in \mathbb{R}$, we study the following problems:

Enumeration. Return the elements $R(P, 1)$, $R(P, 2)$, $\ldots$, $R(P, k)$ in any order.

Reporting. Return the elements in $d(P) \cap (-\infty, r]$ in any order.

Selection. Return the element $R(P, k)$.

Counting. Return the number $K(P, r)$.

The reporting problem is also known as "fixed-radius nearest neighbor search." The counting problem is the same as determining the "rank" of a given distance.

The above formulation of the problems differs slightly from usual convention. For example, the standard selection problem asks for the k-th smallest element $d(p, q)$ over all unordered pair $\{p, q\} \subset P$. The two versions are basically equivalent. Indeed, in most applications, all distances are nonnegative and $d(p, p) = 0$. So, the k-th smallest distance of all unordered pairs is identical to the $(n + 2k)$-th smallest distance of all ordered pairs; in the time bounds, we just replace k with $\Theta(n + k)$.

2.1 From Enumeration to Reporting

It is a straightforward exercise to exhibit an efficient reduction from the reporting problem to the enumeration problem. In this subsection, we show that a general reduction in the reverse direction is possible via random sampling.

Lemma 2.1 *In $O(n)$ time, we can select an r such that $R(P,k) \leq r \leq R(P,k')$ with a positive-constant probability, where $k' = 6(n+k)$.*

Proof: Choose n elements $r_1, \ldots, r_n$ randomly from $d(P)$ and select the m-th smallest element r in the multiset $\{r_1, \ldots, r_n\}$ with $m = 3\lceil k/n \rceil$. It suffices to show that sum of the two probabilities $\Pr\{r < R(P,k)\}$ and $\Pr\{r > R(P,k')\}$ is bounded by a constant strictly less than 1.

If $r < R(P,k)$, then $N = |\{i : r_i < R(P,k)\}|$ is at least m. The random variable N satisfies a binomial distribution with n trials, where each trial has success probability less than k/n^2. The mean $\mu = E[N]$ is less than k/n. Therefore, by Markov's inequality,

$$\Pr\{r < R(P,k)\} \;\leq\; \Pr\{N \geq m\} \;\leq\; \Pr\{N > 3\mu\} \;\leq\; 1/3.$$

If $r > R(P,k')$, then $N' = |\{i : r_i \leq R(P,k')\}|$ is less than m. The random variable N' satisfies a binomial distribution with n trials, where each trial has success probability at least k'/n^2. The mean $\mu' = E[N']$ is at least $k'/n = 6(k/n+1)$. Therefore,

$$\Pr\{r > R(P,k')\} \;\leq\; \Pr\{N' < m\} \;\leq\; \Pr\{N' < \mu'/2\} \;<\; e^{-\mu'/8} \;<\; e^{-3/4} < 1/2,$$

by a standard Chernoff bound (e.g., [30, p. 70]). □

The above idea of taking a random sample of size n to approximate a set of size $O(n^2)$ is hardly new. A similar approach was taken by Matoušek [25] in the context of slope selection and was named "randomized interpolating search." One of the earliest uses of the idea was actually by Rabin when discussing the closest pair problem in his seminal paper on randomized algorithms [32].

Suppose that an algorithm for the reporting problem is available and runs in $T_R(n, K(P,r))$ time. Assume that $T_R(an, b(n+k)) = O(T_R(n,k))$ for any constants a, b. We now apply the above lemma to get a simple randomized algorithm for the enumeration problem.

The possibility of degeneracy (i.e., duplicate distances) creates some technical complications, which we will overcome by introducing an *infinitesimal* δ: its value is assumed to be arbitrarily small but positive. For example, applying the reporting algorithm on $r - \delta$ should produce all distances in the open interval $(-\infty, r)$. (In practice, we can usually avoid treating δ as a variable during the simulation of the algorithm but instead modify the algorithm directly.)

Theorem 2.2 *The enumeration problem can be solved in $O(T_R(n,k))$ expected time.*

Proof: Construct r by Lemma 2.1. Verify that $R(P,k) \leq r \leq R(P,k')$. If not, repeat for a different r; the expected number of trials is bounded by a constant. To verify that $r \geq R(P,k)$, simulate the reporting algorithm on r for $T_R(n,k)$ time steps; the computation should either fail or return a list of k or more distances. To verify that $r \leq R(P,k')$, simulate the reporting algorithm on $r - \delta$ for $T_R(n,k')$ time steps; the computation should succeed with a list of fewer than k' elements.

Now, report all elements in $\Delta = d(P) \cap (-\infty, r)$ and select the first k smallest elements. In case that r is precisely equal to $R(P,k)$, add $\max\{k - |\Delta|, 0\}$ copies of r to the output. As $|\Delta| < k'$, the running time is $O(T_R(n,k'))$. □

2.2 From Selection to Counting

An ordinary binary search reduces the counting problem to the selection problem. In this subsection, we explore the reverse direction. As Agarwal *et al.* [1] noted, a fairly general reduction of selection to counting is possible by parametric search. We point out a simpler approach by randomization.

Our approach is similar to a scheme that Matoušek [25] called "randomized halving" (see also [16]). The idea is a binary search that repeatedly cuts the search interval using a random element in the interval. We describe a recursive procedure to produce such a random element efficiently in general. (Matoušek's procedure is specifically designed for the slope selection problem.)

Suppose that an algorithm is available for the counting problem and runs in time $T_C(n, K(P, r))$. Assume that $T_C(n, k)/n^{\varepsilon}$ is an increasing function in both n and k for some constant $\varepsilon > 0$. Assume also that $T_C(an, b(n + k)) = O(T_C(n, k))$ for constants a, b. As before, degeneracy is handled by using an infinitesimal δ. Note that we can compute $K(P, Q, r)$ using the counting algorithm by the identity

$$K(P, Q, r) \;=\; (K(P \triangle Q, r) + K(P, r) + K(Q, r))/2 - K(P \setminus Q, r) - K(Q \setminus P, r), \tag{1}$$

where $P \triangle Q = (P \setminus Q) \cup (Q \setminus P)$ denotes the symmetric difference.

Lemma 2.3 *Given $A < B$, one can select an element uniformly at random from $d(P, Q) \cap (A, B)$ in $O(T_C(n, k))$ time, where $n = |P| = |Q|$ and $k = K(P \cup Q, B - \delta)$.*

Proof: Define $N(P, Q) = |d(P, Q) \cap (A, B)| = K(P, Q, B - \delta) - K(P, Q, A)$, which can be computed in $O(T_C(n, k))$ time by (1). Partition $P = P_1 \cup P_2$ and $Q = Q_1 \cup Q_2$, with $|P_1|, |P_2|, |Q_1|, |Q_2| \leq \lceil n/2 \rceil$. Compute $N(P_i, Q_j)$ for each i, j. Generate a random pair $(i, j) \in \{1, 2\} \times \{1, 2\}$ such that (i, j) is chosen with probability $N(P_i, Q_j)/N(P, Q)$. Now, recursively select an element uniformly at random from $d(P_i, Q_j) \cap (A, B)$. The running time of this process is $O(T_C(n, k) + T_C(n/2, k) + T_C(n/4, k) + \cdots) = O(T_C(n, k))$. $\square$

Theorem 2.4 *The selection problem can be solved in $O(T_C(n, k) \log n)$ expected time.*

Proof: We will maintain an interval $(A, B]$ containing $R(P, k)$. Initially, we can set $A = 0$ and B with the value r from Lemma 2.1. As in the proof of Theorem 2.2, we can verify that $R(P, k) \leq r \leq R(P, k')$ (this time, by simulating the counting algorithm). If this is not true, repeat for an expected constant number of trials.

Now pick a random element $r \in d(P) \cap (A, B)$ by Lemma 2.3. If no such element exists, then return B. Compute $K(P, r)$ in $T_C(n, k')$ time. If $K(P, r) < k$, then replace A with r. Otherwise, replace B with r.

It suffices to show that the expected number of iterations is logarithmic. Let $N = |d(P) \cap (A, B)|$. Since r is equally likely to be any element in $d(P) \cap (A, B)$, the probability that $|d(P) \cap (A, r)| \geq 2N/3$ and the probability that $|d(P) \cap (r, B)| \geq 2N/3$ are both at most $1/3$. The expected number of iterations to reduce N by a factor of $2/3$ is thus bounded by 3, and consequently, the expected number of iterations to reduce N to 0 is $O(\log n)$. $\square$

Removing the extra $\log n$ factor seems to require stronger oracles than just the counting algorithm.

3 Algorithms for Natural Distances

In this section, we give specific enumeration and selection algorithms for the following kind of distance functions, where objects are points from a fixed-dimensional space $U = \mathbb{R}^d$:

Definition. A distance function $d : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is *natural* if $\| p - q \|_\infty \leq d(p,q) \leq c \| p - q \|_\infty$ for some constant c. Here, $\| \cdot \|_\infty$ denotes the L_∞ norm.

Obviously, all L_p distance functions—in particular, the Euclidean metric—are natural under this definition. The following combinatorial property of Salowe will be important. (The original proof was for the L_∞ metric only, but readjusting constants immediately imply the statement for any natural distance function.)

Proposition 3.1 (Salowe [33]) *If the distance function is natural, then $K(P, ar) = O(|P| + K(P,r))$ for any constant a.*

The enumeration algorithms of Salowe [33] and Lenhof and Smid [24] are based on a lemma similar to the below; a proof is included for completeness. We will use this in our solutions to both the enumeration and the selection problem. (*Pre-sorting* here refers to the computation of the sorted order of the points of P in each of the d coordinates, which can be done in $O(n \log n)$ time.)

Lemma 3.2 *Suppose the distance function is natural. Let $r > 0$ and $P \subset \mathbb{R}^d$ be a given pre-sorted n-point set. In linear time, one can construct subsets $\{P_i\}$ and $\{Q_i\}$ of total size $O(n)$, such that*

(i) *the P_i's are disjoint;*

(ii) *any pair $(p,q) \in P \times P$ with $d(p,q) \leq r$ belongs to some $P_i \times Q_i$;*

(iii) *any pair $(p,q) \in P_i \times Q_i$ has $d(p,q) < 2cr$ for a constant c.*

Proof: We follow the approach of Lenhof and Smid [24] to build a "degraded grid." For each $j \in \{1, \ldots, d\}$, let X_j denote the set of j-th coordinates of the n points. Cover X_j by $\leq n$ disjoint open intervals of length r which we call *j-intervals*. This can be accomplished easily in linear time by scanning the coordinates in sorted order. Define the *neighborhood* of an interval $I = (A, B)$ to be $\widehat{I} = (A - r, B + r)$. Observe that a coordinate can belong to the neighborhoods of at most three j-intervals.

Define a *grid cell* to be a box of the form $h = I_1 \times \cdots \times I_d$ where each I_j is a j-interval. Its *neighborhood* is $\widehat{h} = \widehat{I}_1 \times \cdots \times \widehat{I}_d$. A point can belong to the neighborhoods of at most 3^d grid cells. For each nonempty grid cell h, construct corresponding subsets $P_i = P \cap h$ and $Q_i = P \cap \widehat{h}$. One can verify properties (i)–(iii) easily: if a pair (p,q) has L_∞-distance $\leq r$, then it is in some $P_i \times Q_i$; conversely, any pair in $P_i \times Q_i$ has L_∞-distance $< 2r$.

Regarding the construction time, we can label the j-intervals by integers from $\{1, \ldots, n\}$ and grid cells by d-tuples from $\{1, \ldots, n\}^d$. For each j-th coordinate x, find the label of the j-interval containing x as well as the labels of the j-intervals whose neighborhoods contain x. This can be done by a linear scan. For each point p, we thus have the label of the grid cell containing p and the labels of the grid cells whose neighborhoods contain p. A radix sort then generates the collection of points inside each nonempty grid cell and the collection of points inside its neighborhood. $\square$

3.1 Enumeration

One immediate corollary from Lemma 3.2 is an algorithm for the reporting problem: simply list all the elements in $\Delta = \bigcup_i d(P_i, Q_i)$ and return $\Delta \cap (-\infty, r]$. The running time after pre-sorting is $O(n + |\Delta|) = O(n + K(P, 2cr)) = O(n + K(P, r))$ by Proposition 3.1.

By Theorem 2.2, we have immediately a randomized enumeration algorithm:

Theorem 3.3 *For a pre-sorted n-point set with a natural distance function, the enumeration problem can be solved in $O(n + k)$ expected time.*

Including the time to pre-sort, the expected running time is therefore $O(n \log n + k)$. We now propose a deterministic alternative.

Because of the reporting algorithm, all we need to solve the enumeration problem is a good approximation r to $R(P, k)$. We will obtain such an r by an approximate binary search.

Lemma 3.4 *Under the same setup as in Lemma 3.2, one can reach one of the two conclusions in $O(n)$ time:* (a) $R(P, k) > r$, *or* (b) $R(P, k) < 2cr$.

Proof: Compute $S = \Sigma_i |P_i| \cdot |Q_i|$. If $S < k$, then (a) is true. If $S \geq k$, then (b) is true. $\square$

Lenhof and Smid [24] observed that an approximation of $R(P, k)$ can be found by searching among all $O(n^2)$ L_∞-distances. To avoid generating the entire search space of quadratic size, one performs an implicit binary search, using repeated weighted-median computations like in the techniques of Frederickson and Johnson [20] and Cole [11].

We suggest a simpler approach by observing that an approximation can be found in a reduced search space of linear size. The approximation factor is quite high ($O(n)$), but can be refined easily afterwards. As a result, we bypass the use of a linear-time median-finding subroutine (except at the very end).

Theorem 3.5 *The enumeration problem for a natural distance function can be solved in $O(n \log n + k)$ time.*

Proof: Fix $j \in \{1, \ldots, d\}$. Let $x_1 \leq \cdots \leq x_n$ be the j-th coordinates of the points in sorted order. Define

$$\Xi_j = \{x_2 - x_1,\ x_3 - x_2,\ \ldots,\ x_n - x_{n-1}\}.$$

Construct the set $\Xi = \Xi_1 \cup \cdots \cup \Xi_d$. Let $\xi_1 \leq \cdots \leq \xi_N$ be its elements in sorted order ($N = O(n)$).

We keep indices ℓ and h while maintaining the following invariant:

$$\xi_\ell / 2c \ < \ R(P, k) \ < \ \xi_h.$$

Initially, we can set $\ell = 0$ and $h = N + 1$ with $\xi_0 = 0$ and $\xi_{N+1} = \infty$. By applying Lemma 3.4 with $r = \xi_{\lfloor (\ell + h)/2 \rfloor} / 2c$, we can cut $h - \ell$ by half in $O(n)$ time. Consequently, in $O(n \log n)$ time, we obtain:

$$\xi_\ell / 2c \ < \ R(P, k) \ < \ \xi_{\ell+1}.$$

We know that $\sigma \leq R(P, k) \leq c\sigma$ for some L_∞-distance σ. Since σ is a sum of at most n elements of Ξ, there must exist some ξ_i with $\xi_i \leq \sigma \leq n\xi_i$. It follows that $\xi_i \leq R(P, k) < \xi_{\ell+1}$ and thus $i \leq \ell$. It also follows that $R(P, k) \leq cn\xi_i \leq cn\xi_\ell$.

Next we keep numbers A and B while maintaining the following invariant:

$$A/2c \;<\; R(P,k) \;\leq\; B.$$

Initially, we can set $A = \xi_\ell$ and $B = cn\xi_\ell$. By applying Lemma 3.4 with $r \approx \sqrt{AB}/2c$, we can reduce B/A to approximately $\sqrt{B/A}$ in $O(n)$ time. Consequently, in $O(n \log\log n)$ time, we obtain:

$$A/2c \;<\; R(P,k) \;\leq 2A.$$

Now, we can solve the enumeration problem by reporting all elements in $d(P) \cap [0, 2A]$ and selecting the first k smallest elements, in time $O(n + K(P, 2A)) = O(n + K(P, A/2c)) = O(n + k)$ by Proposition 3.1. □

The above idea also simplifies the parallel version of Lenhof and Smid's enumeration algorithm [24].

We leave as an open problem whether deterministically we can achieve $O(n + k)$ running time for the enumeration problem when the point set is pre-sorted. We do not know, for that matter, a linear-time deterministic algorithm to find the closest pair of a pre-sorted point set; Fortune and Hopcroft [17] showed that $O(n \log\log n)$ time is possible.

3.2 k-Sensitive Selection

We now describe a general strategy to derive k-sensitive bounds for the selection problem when the distance function is natural. We first solve the counting problem and then apply Theorem 2.4.

Theorem 3.6 *Suppose we have an $O(nf(n))$-time algorithm for the counting problem, where f is a concave nondecreasing function. For a natural distance function, we can solve the selection problem in $O(n \log n + nf(k/n)\log n)$ expected time.*

Proof: First note that $K(P, Q, r)$ can be evaluated in $O(nf(n))$ time by (1) if $|P|, |Q| \leq n$. For uneven sizes $|P| = n$ and $|Q| = m$, we can partition Q into subsets $Q_1, \ldots, Q_{\lceil m/n \rceil}$ each of size $\leq n$. Then $K(P, Q, r) = \sum_i K(P, Q_i, r)$ can be evaluated in $O(\lceil m/n \rceil \cdot nf(n)) = O((m + n)f(n))$ time.

A k-sensitive algorithm to compute $k = K(P, r)$ can be obtained from Lemma 3.2:

$$K(P,r) \;=\; \sum_i K(P_i, Q_i, r),$$

where the sizes $n_i = |P_i|$ and $m_i = |Q_i|$ satisfy the bounds

$$\sum_i (m_i + n_i) \;=\; O(n), \quad \text{and} \quad \sum_i m_i n_i \;\leq\; K(P, 2cr) \;=\; O(n + k),$$

by Proposition 3.1. Without loss of generality, assume $m_i \geq n_i$. Then we can calculate all $K(P_i, Q_i, r)$ in time of the order of

$$\sum_i m_i f(n_i) \;\leq\; \left(\sum_i m_i\right) \cdot f\left(\frac{\sum_i m_i n_i}{\sum_i m_i}\right) \;=\; O(nf((n + k)/n)),$$

by Jensen's inequality (since f is concave).

Thus, the counting problem can be solved in $O(n(1+f(K(P,r)/n)))$ time after pre-sorting. The selection problem can then be solved by Theorem 2.4 in $O(n(1+f(k/n))\log n)$ expected time. (Note that the subproblems that occur in the proof of Lemma 2.3 are already pre-sorted.) $\square$

Agarwal *et al.* [1] showed that the counting problem for Euclidean distances in the plane can be solved in $O(n^{4/3}\log^{2/3} n)$ expected time. Applying the above theorem, we have:

Corollary 3.7 *The selection problem for the Euclidean distance function in the plane can be solved in $O(n\log n + n^{2/3}k^{1/3}\log^{5/3} n)$ expected time.*

4 Halfspace Range Counting

Enumerating the k farthest pairs (or the k largest distances) fits into the framework of Section 2 if we simply negate all the distances. Unfortunately, the resulting distance function is no longer natural and the algorithms of Section 3 cannot be applied. The same can be said for variations of the enumeration/selection problem in the bichromatic case (where we take the distance of two points of the same color to be infinite), and in the weighted case (where the actual distance is multiplied by or added to the weight of a point). Therefore, these problems have to be solved by other means.

Take for an example the k farthest pairs problem in the plane under the Euclidean metric. A point (ξ,η) has distance r away from (a,b) if and only if the halfspace $\{(x,y,z) : \xi^2+\eta^2-2\xi x-2\eta y+z\geq r^2\}$ contains the lifted point (a,b,a^2+b^2). Thus, the distance reporting problem for a given $r>0$ can be solved using data structures for halfspace range reporting queries in $\mathbb{R}^3$. By known range searching results [8], we can report the k distances $\geq r$ in $O(n\log n+k)$ time, and by Theorem 2.2, we can enumerate the k farthest pairs in $O(n\log n+k)$ expected time. (For this enumeration problem, Katoh and Iwano [21] described a less efficient algorithm with an $O(n\log n+k^{4/3})$ time bound, and Dickerson and Shugart [14] gave an algorithm for uniformly distributed points with an $O(n\log n + k\log^2 n/\log\log n)$ expected time bound.)

The above approach can also enumerate the k bichromatic closest pairs in the plane. More generally, distance enumeration reduces to halfspace range reporting if it is possible to "linearize" the region $\{q\in U : d(p,q)\leq r\}$ using a fixed number of variables for a given $p\in U$ and $r\in\mathbb{R}$. Similarly, distance selection reduces to halfspace range counting under the same condition using Theorem 2.4.

We now point out a new output-sensitive result for the halfspace range counting problem.

Notation. $f=O^*(g)$ if $f=O(g\cdot n^{\varepsilon})$ for an arbitrarily small constant $\varepsilon>0$.

Theorem 4.1 *Given a parameter $m\geq n$, we can preprocess an n-point set $P\subset\mathbb{R}^d$ in $O^*(m)$ time and space such that the number k of points inside a query halfspace can be determined in time*

$$O^*\left(1+\frac{n}{m^{1/\lfloor d/2\rfloor}}+\left(\frac{n^{\lfloor d/2\rfloor}k^{\lceil d/2\rceil}}{m}\right)^{1/d}\right).$$

The previous query bounds are $O^*(1+n/m^{1/d})$ [10, 26] and $O^*(1+n/m^{1/\lfloor d/2\rfloor}+k)$ [27]. The proof of this theorem is deferred to the appendix. In particular, it implies the following result:

Corollary 4.2 *An online sequence of n halfspace range counting queries on a given n-point set in $\mathbb{R}^d$ require time*

$$O^*(n^{2-2/(\lfloor d/2\rfloor+1)} + (n^{2\lfloor d/2\rfloor}k^{\lceil d/2\rceil})^{1/(d+1)}),$$

where k is the sum of the counts returned by the queries ($k \le n^2$).

Proof: The total cost of answering n queries with counts $k_1, \ldots, k_n$ is asymptotically

$$m + \sum_{i=1}^{n}\left(\frac{n}{m^{1/\lfloor d/2\rfloor}} + \left(\frac{n^{\lfloor d/2\rfloor}k_i^{\lceil d/2\rceil}}{m}\right)^{1/d}\right) \le m + \frac{n^2}{m^{1/\lfloor d/2\rfloor}} + \left(\frac{n^{2\lfloor d/2\rfloor}k^{\lceil d/2\rceil}}{m}\right)^{1/d},$$

because $\sum_i k_i^{\lceil d/2\rceil/d} \le k^{\lceil d/2\rceil/d}n^{\lfloor d/2\rfloor/d}$ by Hölder's inequality. To balance the various terms in the above bound, we set the tradeoff parameter to

$$m = n^{2-2/(\lfloor d/2\rfloor+1)} + (n^{2\lfloor d/2\rfloor}k^{\lceil d/2\rceil})^{1/(d+1)},$$

and the result follows. The value of k is not known in advance though, but a standard trick of "guessing the output size" (e.g., [7]) can easily fix the problem. $\square$

If we want to select the k-th largest distance among n points in the Euclidean plane, then the above result implies an $O^*(n+\sqrt{nk})$ expected time algorithm, since the lifted points have dimension $d = 3$. Improvements could be possible as the actual dimension of the problem is smaller than this "linearization dimension" [3]. We will not go further in this direction here: the main point is that k-sensitive results are obtainable if we just re-examine previous range searching techniques more carefully.

A Appendix: Proof of Theorem 4.1

We describe how halfspace range counting can be solved within the bounds stated in Theorem 4.1. The details are technical and are only sketched here. Basically, the underlying data structures are the same as those of Matoušek [27] for halfspace range reporting. To make query bounds output-sensitive, these are combined with known results on simplex range counting. We first examine the near-linear space case and then consider tradeoffs between space and query time.

Near-Linear Space. Given an n-point set $P \subset \mathbb{R}^d$, a *simplicial partition of size s* is a collection $\{(P_1, \Delta_1), \ldots, (P_s, \Delta_s)\}$ such that (i) the P_i's are disjoint subsets whose union is P, (ii) the Δ_i's are relatively open simplices, and (iii) $P_i \subset \Delta_i$ for each i. The *class size* refers to the maximum of the sizes of $\{P_i\}$. The *crossing number* of a halfspace γ refers to the number of simplices from $\{\Delta_i\}$ that the bounding hyperplane $\partial\gamma$ crosses. (A hyperplane h *crosses* Δ if $h \cap \Delta \ne \emptyset$ and $\Delta \not\subseteq h$.) We say that γ is *ℓ-shallow* if $|P \cap \gamma| \le \ell$.

Lemma A.1 (Matoušek's Partition Theorem for Shallow Hyperplanes [27]) *Given any $1 \le r \le n$, one can find a simplicial partition of size $O(r)$ such that the class size is $O(n/r)$ and the crossing number of any (n/r)-shallow halfspace is $O(r^{1-1/\lfloor d/2\rfloor} + \log r)$. The running time is $O(n)$ if r is a constant.*

A near-linear-space data structure for P can be obtained as follows. Store the cardinality of P and build a linear-space data structure for simplex range counting [10, 26]. Then construct a simplicial partition $\{(P_i, \Delta_i)\}$ using an arbitrarily large constant for r and recursively build data structures for the subsets $\{P_i\}$. The preprocessing time and space of the data structure satisfy the recurrence

$$T(n) \;=\; O(r)\, T(n/r) \;+\; O^*(n),$$

which solves to $T(n) = O^*(n)$.

To count $k = |P \cap \gamma|$ for a query halfspace γ, we determine the simplices $\{\Delta_i\}$ that are crossed by $\partial\gamma$. If the number of such simplices exceeds $C(r^{1-1/\lfloor d/2\rfloor} + \log r)$ for an appropriate constant C (independent of r), then γ is not (n/r)-shallow, and therefore, $k > n/r$. In this case, we will answer the query directly by simplex range counting [10, 26] in time $O^*(n^{1-1/d}) = O^*((rk)^{1-1/d})$. Otherwise, we recursively count $|P_i \cap \gamma|$ for each subset P_i such that Δ_i is crossed by $\partial\gamma$. We can trivially count $|P_i \cap \gamma|$ for each subset P_i with $\Delta_i \subseteq \gamma$. The total count is k.

To analyze the query time, we examine its recursion tree. Let k_ν be the count returned at leaf ν of the tree. We can bound the sum $\sum_\nu k_\nu^{1-1/\lfloor d/2\rfloor}$ by $O^*(n^{1-1/\lfloor d/2\rfloor})$, since the recurrence

$$t(n) \;=\; O(r^{1-1/\lfloor d/2\rfloor} + \log r)\, t(n/r) \;+\; n^{1-1/\lfloor d/2\rfloor}$$

solves to $t(n) = O^*(n^{1-1/\lfloor d/2\rfloor})$. Now, the actual cost of a query is asymptotically bounded by the number of nodes in the recursion tree plus the sum $\sum_\nu k_\nu^{1-1/d}$. The first term is $O^*(n^{1-1/\lfloor d/2\rfloor})$, whereas the second term is bounded by the following using Hölder's inequality:

$$\sum_\nu k_\nu^{1-1/d} \;\le\; \left(\sum_\nu k_\nu^{1-1/\lfloor d/2\rfloor}\right)^{\lfloor d/2\rfloor/d} \left(\sum_\nu k_\nu\right)^{\lceil d/2\rceil/d} \;=\; O^*((n^{1-1/\lfloor d/2\rfloor})^{\lfloor d/2\rfloor/d}\, k^{\lceil d/2\rceil/d}).$$

We conclude that the halfspace range counting problem can be solved using $O^*(n)$ preprocessing time and $O^*(n^{1-1/\lfloor d/2\rfloor} + (n^{\lfloor d/2\rfloor-1} k^{\lceil d/2\rceil})^{1/d})$ query time.

Tradeoffs. Let H be a set of n hyperplanes in $\mathbb{R}^d$. The *level* of a point refers to the number of hyperplanes below it. The *conflict list* $H(\Delta)$ of a simplex Δ consists of all hyperplanes that intersect the interior of Δ. We say that a collection of simplices $\{\Delta_1, \ldots, \Delta_s\}$ is a $(1/r)$-*cutting for the* $(\le \ell)$-*level of size* s if (i) the Δ_i's cover all points of level $\le \ell$, and (ii) $|H(\Delta_i)| \le n/r$ for each Δ_i.

Lemma A.2 (Matoušek's Shallow Cutting Lemma [27]) *Given any $1 \le r \le n$, one can find a $(1/r)$-cutting for the $(\le \ell)$-level of size $O(r^{\lfloor d/2\rfloor} q^{\lceil d/2\rceil})$, where $q = \ell(r/n) + 1$. The running time is $O(n)$ if r is a constant.*

Dualization reduces the halfspace range counting problem to the following: preprocess a set H of n hyperplanes so that given a query point, one can quickly compute its level k. A data structure for processing such queries on H can be obtained as follows. First fix a parameter $p > 0$. If $n \le p$, build the above primal near-linear-space structure. Otherwise, construct a $(1/r)$-cutting $\{\Delta_i\}$ for the $(\le n/r)$-level of size $O(r^{\lfloor d/2\rfloor})$ using an arbitrarily large constant r. Store the number of hyperplanes strictly below each Δ_i, and build a data structure for simplex range counting [10, 26] with

$m(n) = (n/p)^{\lfloor d/2 \rfloor} p$ space. Then recursively build data structures for each $H(\Delta_i)$. The preprocessing time and space satisfy the recurrence

$$T(n) \;=\; \begin{cases} O^*(n) & \text{if } n \le p \\ O(r^{\lfloor d/2 \rfloor})\, T(n/r) \;+\; O^*((n/p)^{\lfloor d/2 \rfloor} p) & \text{if } n > p \end{cases}$$

which solves to $T(n) = O^*(n + (n/p)^{\lfloor d/2 \rfloor} p)$.

We determine the level k of a query point q as follows. If $n \le p$, then use our primal data structure to answer the query in time $O^*(1 + p^{1-1/\lfloor d/2 \rfloor} + (p^{\lfloor d/2 \rfloor - 1} k^{\lceil d/2 \rceil})^{1/d})$. Otherwise, find a simplex Δ_i that contains q. If none exists, then $k > n/r$, and the query can be answered directly by simplex range counting [10, 26] in time $O^*(1 + n/m(n)^{1/d}) = O^*(1 + (n^{\lceil d/2 \rceil} p^{\lfloor d/2 \rfloor - 1})^{1/d}) = O^*(1 + ((rk)^{\lceil d/2 \rceil} p^{\lfloor d/2 \rfloor - 1})^{1/d})$. If we succeed in finding Δ_i, then we recursively compute the level of q with respect to $H(\Delta_i)$. Adding to this the number of hyperplanes strictly below Δ_i, we get k. Since the depth of the recursion is logarithmic, we conclude that the halfspace range counting problem can be solved using $O^*(n + (n/p)^{\lfloor d/2 \rfloor} p)$ preprocessing time and

$$O^*(1 + p^{1-1/\lfloor d/2 \rfloor} + (p^{\lfloor d/2 \rfloor - 1} k^{\lceil d/2 \rceil})^{1/d})$$

query time. Finally setting $p^{\lfloor d/2 \rfloor - 1} = n^{\lfloor d/2 \rfloor}/m$ yields the bounds stated in Theorem 4.1.

References

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

[2] P. K. Agarwal and J. Erickson. Geometric range searching and its relatives. To appear in *Discrete and Computational Geometry: Ten Years Later* (B. Chazelle, J. E. Goodman, and R. Pollack, ed.), AMS Press.

[3] P. K. Agarwal and J. Matoušek. On range searching with semialgebraic sets. *Discrete Comput. Geom.*, 11:393–418, 1994.

[4] S. Arya and M. Smid. Efficient construction of a bounded-degree spanner with low weight. *Algorithmica*, 17:33–54, 1997.

[5] J. L. Bentley and M.I. Shamos. Divide-and-conquer in multidimensional space. In *Proc. 8th ACM Sympos. Theory Comput.*, pages 220–230, 1976.

[6] M. de Berg and O. Schwarzkopf. Cuttings and applications. *Int. J. Comput. Geom. Appl.*, 5:343–355, 1995.

[7] T. M. Chan. Output-sensitive results on convex hulls, extreme points, and related problems. *Discrete Comput. Geom.*, 16:369–387, 1996.

[8] T. M. Chan. Random sampling, halfspace range reporting, and construction of ($\le k$)-levels in three dimensions. In *Proc. 39th IEEE Sympos. Found. Comput. Sci.*, pages 586–595, 1998.

[9] B. Chazelle, L. Guibas, and D. T. Lee. The power of geometric duality. *BIT*, 25:76–90, 1985.

[10] B. Chazelle, M. Sharir, and E. Welzl. Quasi-optimal upper bounds for simplex range searching and new zone theorems. *Algorithmica*, 8:407–429, 1992.

[11] R. Cole. Slowing down sorting networks to obtain faster sorting algorithms. *J. ACM*, 34:200–208, 1987.

[12] M. T. Dickerson, R. L. S. Drysdale, and J.-R. Sack. Simple algorithms for enumerating interpoint distances and finding k nearest neighbors. *Int. J. Comput. Geom. Appl.*, 2:221–239, 1992.

[13] M. T. Dickerson and D. Eppstein. Algorithms for proximity problems in higher dimensions. *Comput. Geom. Theory Appl.*, 5:277–291, 1996.

[14] M. T. Dickerson and J. Shugart. A simple algorithm for enumerating longest distances in the plane. *Inform. Process. Lett.*, 45:269–274, 1993.

[15] M. Dietzfelbinger, T. Hagerup, J. Katajainen, and M. Penttonen. A reliable randomized algorithm for the closest-pair problem. *J. Algorithms*, 25:19–51, 1997.

[16] M. Dillencourt, D. Mount, and N. Netanyahu. A randomized algorithm for slope selection. *Int. J. Comput. Geom. Appl.*, 2:1–27, 1992.

[17] S. J. Fortune and J. E. Hopcroft. A note on Rabin's nearest-neighbor algorithm. *Inform. Process. Lett.*, 8:20–23, 1979.

[18] M. Golin, R. Raman, C. Schwarz, and M. Smid. Simple randomized algorithms for closest pair problems. *Nordic J. Comput.*, 2:3–27, 1995.

[19] M. T. Goodrich. Geometric partitioning made easier, even in parallel. In *Proc. 9th ACM Sympos. Comput. Geom.*, pages 73–82, 1993.

[20] G. N. Frederickson and D. B. Johnson. The complexity of selection and ranking in $X + Y$ and matrices with sorted rows and columns. *J. Comput. Sys. Sci.*, 24:197–208, 1982.

[21] N. Katoh and K. Iwano. Finding k farthest pairs and k closest/farthest bichromatic pairs for points in the plane. *Int. J. Comput. Geom. Appl.*, 5:37–51, 1995.

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

[23] S. Khuller and Y. Matias. A simple randomized sieve algorithm for the closest-pair problem. *Inform. Comput.*, 118:34–37, 1995.

[24] H.-P. Lenhof and M. Smid. Sequential and parallel algorithms for the k closest pairs problem. *Int. J. Comput. Geom. Appl.*, 5:273–288, 1995.

[25] J. Matoušek. Randomized optimal algorithm for slope selection. *Inform. Process. Lett.*, 39:183–187, 1991.

[26] J. Matoušek. Efficient partition trees. *Discrete Comput. Geom.*, 8:315–334, 1992.

[27] J. Matoušek. Reporting points in halfspaces. *Comput. Geom. Theory Appl.*, 2:169–186, 1992.

[28] J. Matoušek. Geometric range searching. *ACM Comput. Surveys*, 26:421–461, 1994.

[29] N. Megiddo. Applying parallel computation algorithms in the design of serial algorithms. *J. ACM*, 30:852–865, 1983.

[30] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, New York, 1995.

[31] M. Pellegrini. On counting pairs of intersecting segments and off-line triangle range searching. *Algorithmica*, 17:380–398, 1997.

[32] M. O. Rabin. Probabilistic algorithms. In *Algorithms and Complexity* (J. F. Traub, ed.), pages 21–30, Academic Press, New York, 1976.

[33] J. S. Salowe. Enumerating interdistances in space. *Int. J. Comput. Geom. Appl.*, 2:49–59, 1992.

[34] M. Smid. Closest-point problems in computational geometry. To appear in *Handbook of Computational Geometry* (J. Urrutia and J. Sack, ed.), North-Holland.

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