Shallow Interdistance Selection and Interdistance Enumeration

J. S. Salowe

Computer Science Report No. TR-91-01
January 11, 1991

Shallow Interdistance Selection and Interdistance Enumeration

by

J.S. Salowe

Department of Computer Science
University of Virginia
Charlottesville, Virginia 22903

ABSTRACT

Shallow interdistance selection refers to the problem of selecting the k^{th} smallest interdistance, $k \leq n$, from among the $\binom{n}{2}$ interdistances determined by a set of n points in $\Re^d$. Shallow interdistance selection has a concrete application — it is a crucial component in the design of a data structure that dynamically maintains the minimum interdistance in $O(\sqrt{n}\log n)$ time per operation (Smid [6]). In addition, the study of shallow interdistance selection may provide insight into developing more efficient algorithms for the problem of selecting Euclidean distances (Agarwal *et al.* [1]). We give a shallow interdistance selection algorithm which takes optimal $O(n \log n)$ time and works in any L_p metric. To do this, we prove two interesting related results. The first is a combinatorial result relating the rank of x to the rank of $2x$. The second is an algorithm which enumerates all pairs of points within interdistance x in time proportional to the rank of x (plus $O(n \log n)$).

1. Introduction

We consider the problem of selecting the k^{th} smallest interdistance, $k \leq n$, from among the $\binom{n}{2}$ determined by a set P of n points in $\Re^d$. This problem is in part motivated by recent progress by Smid [6] in the design of an efficient, linear-sized data structure supporting the following three operations:

insert (x,P): form point set $P \cup \{x\}, x \in \Re^d$;

delete (x,P): form point set $P - \{x\}$;

minimum (P): return a pair of points having minimum interdistance with respect to a given metric.

Smid's data structure supports these three operations in $O(n^{2/3}\log n)$ time if the interpoint distance is given by an L_p metric. Smid's algorithm depends on an algorithm for "shallow" interdistance selection. Specifically, Smid showed how to find the ordered sequence of the $O(n^{2/3})$ smallest distances determined by P in $O(n \log n)$ time, and he noted that the update times for his data structure could be improved to $O(\sqrt{n}\log n)$ time if the ordered sequence of n smallest interdistances could be found in $O(n \log n)$ time. (We note that Smid has recently developed a second dynamic algorithm which has polylogarithmic update times but uses $O(n \log^{O(1)} n)$ storage [5], so the actual impact of shallow interdistance selection on closest point problems is primarily of theoretical interest.)

A second motivation to study this problem is to determine the true complexity of interdistance selection. Salowe [4] presented an $O(n \log^d n)$ time algorithm to select L_∞ interdistances in d-dimensional space. L_1 interdistances in the plane may be selected in $O(n \log^2 n)$ time. Recently, Agarwal, Aronov, Sharir and Suri [1] gave an $O(n^{3/2}\log^{5/2} n)$ time algorithm for L_2 interdistances in the plane. The best result for points in d-dimensional space (with respect to the Euclidean metric) is due to Chazelle [2], who devised a nearly-quadratic algorithm based on Yao's technique [8] for constructing minimum spanning trees in d-dimensional space. What are the true bounds? What is the relationship between the apparently easy selection in the L_∞ metric to the seemingly harder selection in the L_2 metric?

In this paper, we devise an $O(n \log n)$ time algorithm to select the k^{th} smallest interdistance determined by P, where $k \leq n$. This algorithm works for any L_p metric and is optimal; it relies on an interesting combinatorial result and an algorithm which enumerates interdistances less than or equal to x in time proportional to the rank of x (plus $O(n \log n)$). The organization is the following. In Section 2, we state and prove the combinatorial result. Section 3 contains the interdistance selection algorithm. The first subsection describes an algorithm for "interdistance enumeration," the second subsection presents an optimal shallow interdistance selection algorithm for the L_∞ metric, the third subsection presents an optimal algorithm for L_p metrics. Some remarks are made in Section 4.

2. Combinatorial Results

We begin with a question involving interdistance ranks. Given point set P with distances measured in the L_p metric, the rank $r_p(x)$ of distance x is $\#\{(u,v) : d_p(u,v) \leq x,\ u,v \in P\ ,\ u \neq v\ \}$. Here, (u,v) is considered to be an unordered pair so that (u,v) and (v,u) are not double-counted. Note that the set of points within distance x of point u in the L_∞ metric are contained in a *hypercube* centered at u of dimension (or *side length*) $2x$. Suppose that distance x has rank k. What is the rank of distance $2x$? It is easy to describe a point set in the plane where distance x is the smallest interdistance, but distance $2x$ has rank $\Omega(n)$ (See Figure 1). In general, however, the ranks of x and $2x$ do not differ

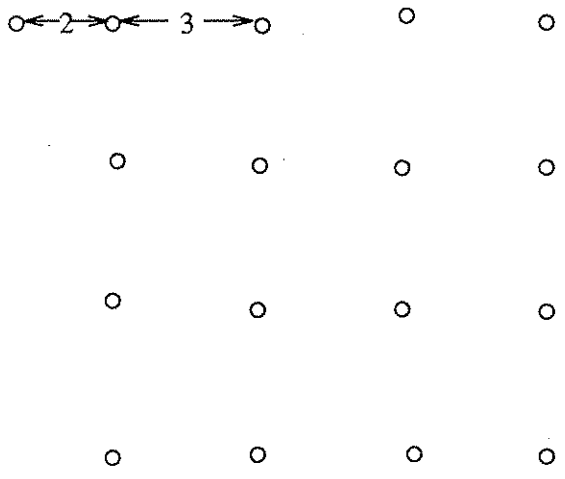

Figure 1

significantly.

Theorem 1: Let P be a set of n points in $\Re^d$; then if $r_\infty(x) = k$, $k \le r_\infty(2x) \le c(d)(k+n)$, where $c(d) = 5^{2d}$.

Proof: The lower bound is obvious. For the upper bound, divide up $\Re^d$ into hypercubes of side x. Note that any two points inside a given hypercube are within distance x of each other. Label the non-empty hypercubes arbitrarily with integers 1, 2,.... Let n_i be the number of points of P inside the i^{th} hypercube. Then

$$k \ge \sum_i \binom{n_i}{2}.$$

Now consider distance $2x$. With respect to hypercube i, the points within distance $2x$ of hypercube i are contained in a larger hypercube C_i of side $5x$ centered at the same point as the center of hypercube i. There are at most 5^d original hypercubes intersecting C_i, and an upper bound on the number of interdistances less than or equal to $2x$ involving points in C_i is:

$$B_i = \sum_{j \cap C_i} n_i n_j$$

Among these hypercubes having non-empty intersection with C_i, charge $5^d n_k^2$ to the hypercube k containing the most points. Note that $B_i \le 5^d n_k^2$.

Summing up all charges, no hypercube can be charged more than 5^d times, so

$$\sum_i B_i \le 5^{2d} \sum_i n_i^2.$$

However,

$$\sum_i n_i^2 = n + 2\sum_i \binom{n_i}{2}.$$

Recalling that distances are double-counted, we have

$$r_\infty(2x) \le \frac{1}{2} \sum_i B_i \le \frac{1}{2} 5^{2d} \sum_i n_i^2$$

$$= \frac{1}{2} 5^{2d} (n + 2\sum_i \binom{n_i}{2})$$

$$\le 5^{2d}(n+k).$$

□

Recall that the L_p metrics are related by constant factors; that is if u and v are points in $\Re^d$, $d_\infty(u,v) \le d_p(u,v) \le d \cdot d_\infty(u,v)$.

Corollary 1:

$$r_\infty(tx) \le c(d)^{\lceil \lg t \rceil}\, r_\infty(x) + n\,\frac{c(d)^{\lceil \lg t \rceil + 1} - c(d)}{c(d) - 1},$$

where $t \ge 1$ and $c(d) = 5^{2d}$.

Proof: We first note that $r_\infty(tx) \le r_\infty(2^{\lceil \lg t \rceil} x)$ since ranks are monotonically nondecreasing. We now show by induction on i that

$$r_\infty(2^i x) \le c(d)^i\, r_\infty(x) + n\,\frac{c(d)^{i+1} - c(d)}{c(d)-1}$$

for $i \ge 1$. Theorem 1 proves the basis. In general,

$$r_\infty(2^i x) \le c(d)(r_\infty(2^{i-1} x) + n).$$

Applying the inductive hypothesis,

$$r_\infty(2^i x) \le c(d)((c(d)^{i-1}\, r_\infty(x) + n\,\frac{c(d)^i - c(d)}{c(d) - 1}) + n).$$

Therefore,

$$r_\infty(2^i x) \le c(d)^i\, r_\infty(x) + n\,\frac{c(d)^{i+1} - c(d)}{c(d) - 1},$$

as asserted. □

Corollary 2: Let $y = x/d$. Then

$$r_\infty(y) \le r_p(x) \le c(d)^{\lceil \lg d \rceil} r_\infty(y) + n\,\frac{c(d)^{\lceil \lg d \rceil + 1} - c(d)}{c(d) - 1}.$$

Proof: Recall that $r_p(x) = \#\{(u,v) : d_p(u,v) \le x\}$. Since $d_\infty(u,v) \le d_p(u,v)$, $r_p(x) \le \#\{(u,v) : d_\infty(u,v) \le x\}$, so $r_p(x) \le r_\infty(x)$. Since $d_p(u,v) \le d \cdot d_\infty(u,v)$ and $r_\infty(x/d) = \#\{(u,v) : d_\infty(u,v) \le x/d\}$, implying $r_\infty(x/d) = \#\{(u,v) : d \cdot d_\infty(u,v) \le x\}$ and $r_\infty(x/d) \le \#\{(u,v) : d_p(u,v) \le x\}$. As a consequence, $r_\infty(x/d) \le r_p(x)$.

Now let $y = x/d$. We now have $r_\infty(y) \le r_p(x) \le r_\infty(d \cdot y)$, and we get the stated result by applying Corollary 1. □

Corollary 2 states that if y is the n^{th} smallest interdistance with respect to the L_∞ metric, $d \cdot y$ has rank $\Theta(n)$ with respect to any L_p metric.

3. Shallow Selection Algorithms

Using Corollary 2, we give an efficient shallow selection algorithm based on Salowe's L_∞ interdistance selection algorithm [4]. The algorithm is the following:

$SELECT_p(P,k)$
{ P is the input set, $k \le n$ is the rank, p indicates the metric. }

1. Preprocess P for orthogonal range queries using the layered range tree [3]
2. $y = SELECT_\infty(P,n)$
3. $L = \emptyset$
   For each $u \in P$
   $L = L \cup Range\text{-}query\text{-}report(P,u,d \cdot y)$
   { *Range-query-report* returns those points inside the orthogonal hypercube of dimension $2d \cdot y$ centered at u. }
4. Select $2k^{th}$ smallest element in L
   { Note that interdistances are double counted. }

Step 1 takes $O(n \log^{d-1} n)$ time [3], and step 2 takes $O(n \log^d n)$ time [4]. Step 3 takes $O(n \log^{d-1} n + n)$ time, since Corollary 2 states that $\Theta(n)$ points are reported. Step 4 takes $O(n)$ time by the linear-time selection algorithm. Therefore, the algorithm takes $O(n \log^d n)$ time. Note that the actual k smallest interdistances all appear on L.

We now show how to speed up this algorithm to obtain an optimal $O(n \log n)$ time algorithm for shallow selection. Optimality is justified by the observation that the time required to find the smallest interdistance is $\Omega(n \log n)$ in the algebraic decision tree model [3].

The improvement is based on an efficient algorithm which, given x, reports all pairs of points within L_∞ distance x of each other. After $O(n \log n)$ preprocessing, the algorithm returns all pairs whose interdistance is at most x in $O(n + r_\infty(x))$ time. In fact, we can modify the algorithm to do a bit more — after preprocessing, the algorithm can report in $O(n)$ time that $r_\infty(x) > c \cdot n$ for some constant c. This algorithm, the *interdistance enumeration algorithm*, is used to speed up steps 1, 2, and 3 above.

3.1. The Interdistance Enumeration Algorithm

The interdistance enumeration algorithm is inspired by Vaidya's optimal all-nearest-neighbors algorithm; we assume the reader is familiar with Vaidya's paper [7], though we briefly sketch the relevant results to be self-contained. (Terminology is adapted from Vaidya's paper.)

Given a set P of n points in $\Re^d$, Vaidya's algorithm finds a nearest neighbor to each point in P. The algorithm makes use of hypercubes called "boxes." During the course of the algorithm, P is partitioned by a set of disjoint boxes B called a *box list*. Initially, B consists of a single box b_0 which is a smallest box containing P. During a *stage*, a largest box in B, say b, is subdivided by d mutually orthogonal hyperplanes passing through its center. The resulting 2^d boxes $\overline{b}_1, \ldots, \overline{b}_{2^d}$ make up the set $immediate\text{-}successors(b)$. Discarding the empty immediate successors and shrinking the remaining boxes as much as possible without changing the contents, one arrives at a set of boxes $b_1, \ldots, b_j$ called $successors(b)$. B then becomes $B \cup successors(b) - \{b\}$. This process stops when each box in B contains exactly one point from P. Vaidya associates two sets with each box; these sets are irrelevant for the application considered here.

The interdistance enumeration algorithm consists of three steps. Step 1, the box-subdivision step described above, takes $O(n \log n)$ time [7]. The result is a "tree-of-boxes" containing at most $2n - 1$ boxes rooted at b_0 with the property that the children of box b are precisely *successors*(b). Constructing this tree-of-boxes is the preprocessing step for the enumeration algorithm. Along with each box b in the tree, we associate the quantities $d_{\max}(b) = \max \{ d_\infty(u,v) : u,v \in b \}$ and $n(b) = \# b \cap P$. The boxes are also labeled in the order they were subdivided. The points $b \cap P$ can be found in time proportional to $n(b)$ by traversing the subtree rooted at b (points are contained in 0-volume boxes which are leaves in the tree-of-boxes).

Step 2 finds a specific partition of the boxes and determines pairs of these boxes within interdistance x of each other, where x is the input distance. Specifically, the tree-of-boxes obtained from step 1 is traversed in the order the boxes were subdivided, and the sequence of box lists is re-created. Let $d_{\min}(b,b') = \min \{d_\infty(u,v) : u \in b, v \in b'\}$. Let B be the box list just before $\bar{b}$, the largest box in B, is subdivided. With each box b on the box list B, the set of boxes $B'(b) \subseteq B$ for which $d_{\min}(b,b') \le x$, $b' \in B'(b)$, is maintained. After $\bar{b}$ is subdivided to created *successors*$(\bar{b})$, the set $B'(\bar{b})$ is examined to update the remaining sets $B'(b)$ to reflect $B = B \cup$ *successors*$(\bar{b}) - \{\bar{b}\}$. Specifically, a new list $B(b_i)$ is created for each $b_i \in$ *successors*$(\bar{b})$, and $\bar{b}$ is replaced by an appropriate subset of *successors*$(\bar{b})$ for each b such that $b \in B'(\bar{b})$. This process stops the first time a box to be subdivided has side length less than or equal to x. (Implementation details are left to the reader.) We show below that at most a constant number of changes are made to B and all the $B'(b)$, so step 2 takes $O(n)$ time.

Step 3 uses the output from step 2 to actually compute the pairs of points having interdistance x or less. At the end of step 2, there is a box list B consisting solely of boxes of side length less than or equal to x, and, for each box $b \in B$, there is a list $B'(b)$ containing all of the boxes in B within distance x of a point in b. The L_∞ rank of x is

$$\sum_{b \in B} \binom{n(b)}{2} + 1/2 \sum_{b \in B} \sum_{b' \in B'(b)} \#\{u,v : u \in b \cap P, v \in b \cap P,\ d_\infty(u,v) \le x\}.$$

It is clear that the rank and the pairs of points within x of each other can be computed from B and the $B'(b)$ in time

$$\sum_{b \in B} \binom{n(b)}{2} + \sum_{b \in B} \sum_{b' \in B'(b)} n(b)\, n(b').$$

We show that this term is $O(n + r_\infty(x))$; this is the cost of step 3.

We first prove that step 2 takes $O(n)$ time. The proof of the following lemma appears in Vaidya [7] where it is asserted for any L_p metric, so the statement is a bit weaker than one specific to the L_∞ metric.

Lemma 1: [Paraphrased from Vaidya, Packing Lemma 1] Let r be a positive integer. Let b be a largest box in B. Then the number of boxes b' in B such that $d_{\min}(b,b') \le r d_{\max}(b)$ is at most $2^d(2rd+3)^d$.

A corollary to Lemma 1 is the following:

Corollary 3: Let b be a largest box in B which is to be subdivided in step 2. The number of boxes b' in B such that $d_{\min}(b,b') \le x$ is at most $2^d(2d+3)^d$.

Proof: Let $r = 1$ and recall that $d_{\max}(b) > x$. □

In step 2, since $O(n)$ boxes are subdivided and there is a bound on the number of changes made to the $B'(b)$ sets, the $O(n)$ time bound follows.

The time bound for step 3 is a consequence of Corollary 1. All boxes in B have side length at most x, so if $b' \in B'(b)$, then any point in b' must be within distance $3x$ from any point in b. This is because $d_{\max}(b) \le x$, $d_{\max}(b') \le x$, and $d_{\min}(b,b') \le x$. Therefore,

$$\sum_{b \in B} \binom{n(b)}{2} + \sum_{b \in B} \sum_{b' \in B'(b)} n(b)n(b').$$

is bounded by $2r_\infty(3x)$, and Corollary 1 implies $r_\infty(3x)$ is $O(n + r_\infty(x))$.

Theorem 2: Let P be a set of n points in $\Re^d$, and let x be positive. The the L_∞ interdistances of length at most x can be found in $O(n \log n + r_\infty(x))$ time.

Using Corollary 2, we can easily generalize Theorem 2 to L_p metrics.

How do we use the interdistance enumeration algorithm for shallow selection? Suppose the preprocessing step, step 1, has already been done. If $r_\infty(x)$ is $O(n)$, then the rest of the interdistance enumeration algorithm takes $O(n)$ time. However, if $r_\infty(x)$ is large, say greater than $c \cdot n$ for some c, we can force the interdistance enumeration algorithm to stop in $O(n)$ time and report "too large" once the bound on $2r_\infty(3x)$ is exceeded in step 3.

3.2. Shallow L-Infinity Interdistances

Salowe's L_∞ interdistance selection algorithm uses a technique called parametric search [4]. The reader is referred to that paper for details. In Salowe's algorithm, a sequentialized parallel sorting algorithm determines $O(\log n)$ distances to be ranked, each in $O(n \log^{d-1} n)$ time. In fact, the actual rank of a distance is not needed. Instead, one needs the relationship between distance x and the unknown distance x^* with rank k. (Is $x < x^*$, is $x = x^*$, or is $x > x^*$?) For shallow selection, we can therefore make use of the "too large" feature explained above.

Specifically, after $O(n \log n)$ preprocessing, we can select a shallow interdistance in $O(n \log n)$ time using parametric search and the interdistance enumeration algorithm. Suppose we seek an interdistance with rank $k \le n$. $O(\log n)$ distances x_i are ranked. If $r_\infty(x_i) > n$, then the interdistance enumeration algorithm stops in $O(n)$ time and reports "too large." If $r_\infty(x_i) \le n$, then the interdistance enumeration algorithm computes and counts the pairs of points within interdistance x, giving the exact rank. The total time expended is therefore $O(n \log n)$, and we have:

Lemma 2: Given a set P of n points in $\Re^d$, the k^{th} smallest L_∞ interdistance can be selected in $O(n \log n)$ time, $k \le n$.

3.3. Shallow L-p Interdistances

An efficient algorithm to select shallow L_p interdistances is the following.

$SELECT_p(P,k)$
{ P is the input set, n is the rank, p indicates the metric. }

1. Perform Step 1 of the Interdistance Enumeration Algorithm
2. $y = SELECT_\infty(P,n)$
3. Perform Steps 2 and 3 of the Interdistance Enumeration Algorithm for distance $d \cdot y$. Call the reported pairs L.
4. Select k^{th} smallest interdistance in L

Vaidya's analysis and Lemma 2 imply that steps 1 and 2 take $O(n)$ time. Corollary 2 and Theorem 2 imply that step 3 takes $O(n)$ time and the size of L is $O(n)$. The linear-time selection algorithm proves the rest, giving:

Theorem 3: Given a set P of n points in $\Re^d$, the k^{th} smallest L_p interdistance can be selected in $O(n \log n)$ time, $k \leq n$.

As an immediate corollary, we improve on Smid's result:

Corollary 4: There is a linear-sized data structure for point sets which supports *insert*, *delete*, and *minimum* in $O(\sqrt{n} \log n)$ time per update.

4. Remarks

We have presented an algorithm which, given n points in $\Re^d$, selects the k^{th} smallest interdistance, $k \leq n$, in optimal $O(n \log n)$ time. The algorithm works in any L_p metric. To obtain the algorithm, we proved a combinatorial result interrelating distances and ranks. We also described a nontrivial algorithm for the problem of enumerating all interdistances less than or equal to x in time proportional to the rank of x (including an $O(n \log n)$ term).

There are several interesting questions. First, what is the complexity of selecting the median? Second, can a result analogous to the one above be obtained when selecting the k^{th} largest interdistance?

5. References

1. P. K. Agarwal, B. Aronov, M. Sharir and S. Suri, Selecting Distances in the Plane, *Sixth ACM Symposium on Computational Geometry*, 1990, pp. 321-331.

2. B. Chazelle, New Techniques for Computing Order Statistics in Euclidean Space, *First ACM Symposium on Computational Geometry*, 1985, pp. 125-134.

3. F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*, Springer Verlag, New York, NY, 1985.

4. J. S. Salowe, L-Infinity Interdistance Selection by Parametric Search, *Inf. Proc. Letters*, **30**, 1989, pp. 9-14.

5. M. Smid, Maintaining the Minimal Distance of a Point Set in Polylogarithmic Time, *Universitat des Saarlandes 13/90*, 1990.

6. M. Smid, Maintaining the Minimal Distance of a Point Set in Less Than Linear Time, *Universitat des Saarlandes 06/90*, 1990.

7. P. M. Vaidya, An $O(n \log n)$ Algorithm for the All-Nearest-Neighbors Problem, *Discrete Comput. Geom.*, **4**, 1989, pp. 101-115.

8. A. C. Yao, On Constructing Minimum Spanning Trees in k-Dimensional Spaces and Related Problems, *Siam J. on Computing*, **11**, 1982, pp. 721-736.