The k closest pairs problem

Hans-Peter Lenhof Michiel Smid

Max-Planck-Institut für Informatik

W-6600 Saarbrücken, Germany

April 1992

Abstract

We give an algorithm that computes the k closest pairs in a set of n points in d-dimensional space in $O(n \log n + k \log n \log(n^2/k))$ time.

1 Introduction

In a recent paper, Katoh and Iwano [2] give a technique for solving problems such as finding the k furthest pairs in a set of n planar points, and finding the k closest bichromatic pairs in a set of n red and n blue points in the plane.

In this note, we show that their method can also be applied to find the k closest pairs in set of n points in d-space.

For this problem, the following results are known. Smid [4] shows how to compute the $n^{2/3}$ closest pairs in $O(n \log n)$ time. This result was extended in two directions. First, for the planar case, Dickerson and Drysdale [1] show how to compute the k closest pairs—ordered by their distances—in $O(n \log n + k \log n)$ time. Second, Salowe [3] gives an algorithm that computes the n closest pairs in $O(n \log n)$ time. The latter result holds for an arbitrary dimension.

Let S be a set of n points in d-space and let k be an integer such that $1 \leq k \leq \binom{n}{2}$. We give a recursive algorithm that computes the k closest pairs in S. Distances are measured in an arbitrary L_t-metric, where $1 \leq t \leq \infty$.

The algorithm works as follows. If $n^2 \leq 4k$, then compare all pairs of points in S and output the k closest ones.

Assume that $n^2 > 4k$. Let $l := \lceil 4k/n \rceil$. Use Vaidya's algorithm [5] to compute for each point in S its l nearest neighbors. This gives a list of ln pairs of points. In this list, select the $2k$ closest pairs. This gives a multiset D^+ of size $2k$. (Pairs may be represented twice in D^+.) Let D be the set obtained from D^+ by removing duplicates. Then, $k \leq |D| \leq 2k$.

Note that—in general—D does not contain all k closest pairs of S. Therefore, let

$$S' := \{p \in S : (p,q) \in D^+ \text{ for all } l \text{ nearest neighbors } q \text{ of } p\}.$$

Use the same algorithm to find the k closest pairs in the set S'. Let D' be the list containing these pairs.

Then, in the final step, select the k closest pairs in the set $D \cup D'$.

This is the entire algorithm. We first prove the correctness. Then, we analyze the running time.

Lemma 1 *The set $D \cup D'$ contains all k closest pairs in S. As a result, the algorithm is correct.*

Proof: Let $\{p,q\}$ be one of the k closest pairs. We distinguish three cases.
Case 1: q is one of the l nearest neighbors of p. Then, $\{p,q\} \in D$.
Case 2: p is one of the l nearest neighbors of q. Then, $\{p,q\} \in D$.
Case 3: q is not one of the l nearest neighbors of p, and p is not one of the l nearest neighbors of q.

Let $1 \leq l' \leq l$ and let r be the l'-th nearest neighbor of p. Let d_k be the k-th smallest distance in S. Then

$$d(p,r) \leq d(p,q) \leq d_k.$$

It follows from the definition of D^+ that it must contain the pair (p,r). Since l' is arbitrary, it follows that $p \in S'$.

By a symmetric argument, it follows that $q \in S'$. Since $S' \subseteq S$, the pair (p,q) must be one of the k closest pairs in S'. Therefore, $\{p,q\} \in D'$. ∎

To analyze the running time, we need the following lemma.

Lemma 2 *The set S' has size at most $n/2$.*

Proof: Clearly, $|D^+| \geq l\,|S'|$. Since $|D^+| = 2k$, we get

$$|S'| \leq \frac{2k}{l} = \frac{2k}{\lceil 4k/n \rceil} \leq \frac{2k}{4k/n} = n/2. \quad \blacksquare$$

Let $T(n,k)$ denote the running time of the algorithm. Since Vaidya's algorithm takes $O(ln \log n)$ time, it follows that for $n^2 > 4k$,

$$\begin{aligned} T(n,k) &= O(ln \log n + k \log k) + T(|S'|, k) \\ &= O((n+k)\log n) + T(n/2, k). \end{aligned}$$

Clearly, $T(n,k) = O(n^2)$ for $n^2 \leq 4k$.

Applying the recursive relation repeatedly, we obtain (assuming the constant in the Big-O-bound is one)

$$\begin{aligned} T(n,k) &\leq \sum_{i=0}^{j} \left(\frac{n}{2^i} + k\right) \log \frac{n}{2^i} \; + \; T(n/2^{j+1}, k) \\ &\leq (n + jk)\log n + T(n/2^{j+1}, k). \end{aligned}$$

Now take $j = \lfloor \frac{1}{2} \log(n^2/k) \rfloor$. (Note that $j \geq 1$, because $n^2 > 4k$.) Then

$$\frac{n}{2^{j+1}} \leq \frac{n}{\sqrt{n^2/k}} = \sqrt{k},$$

and, therefore,

$$T(n/2^{j+1}, k) \leq T(\sqrt{k}, k) = O(k).$$

This shows that

$$T(n,k) = O(n \log n + k \log n \log(n^2/k)).$$

We have proved the following theorem.

Theorem 1 *Let S be a set of n points in d-space and let $1 \leq k \leq \binom{n}{2}$. We can find the k closest pairs in the set S in $O(n \log n + k \log n \log(n^2/k))$ time.*

We can obtain the k closest pairs ordered by their distances, by using the above algorithm and then sorting the list of k closest pairs. This adds $O(k \log k)$ to the time complexity, which is less than the time bound of our algorithm. Hence, we can also solve the ordered k closest pairs problem in $O(n \log n + k \log n \log(n^2/k))$ time.

References

[1] M.T. Dickerson and R.S. Drysdale. *Enumerating k distances for n points in the plane.* Proc. 7-th ACM Symp. on Comp. Geom., 1991, pp. 234-238.

[2] N. Katoh and K. Iwano. *Finding k farthest pairs and k closest/farthest bichromatic pairs for points in the plane.* Proc. 8-th ACM Symp. on Comp. Geom., 1992, to appear.

[3] J.S. Salowe. *Shallow interdistance selection and interdistance enumeration.* Proc. WADS'91, Lecture Notes in Computer Science, Vol. 519, Springer-Verlag, Berlin, 1991, pp. 117-128.

[4] M. Smid. *Maintaining the minimal distance of a point set in less than linear time.* Algorithms Review **2** (1991), pp. 33-44.

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