DIVIDE-AND-CONQUER IN MULTIDIMENSIONAL SPACE

Jon Louis Bentley
Department of Computer Science
University of North Carolina
Chapel Hill, NC 27514

and

Michael Ian Shamos
Departments of Computer Science and Mathematics
Carnegie-Mellon University
Pittsburgh, PA 15213

Abstract

We investigate a divide-and-conquer technique in multidimensional space which decomposes a geometric problem on N points in k dimensions into two problems on N/2 points in k dimensions plus a single problem on N points in k-1 dimension. Special structure of the subproblems is exploited to obtain an algorithm for finding the two closest of N points in O(N log N) time in any dimension. Related results are discussed, along with some conjectures and unsolved geometric problems.

Introduction

A principal failing of computational geometry [Shamos, Shamos and Hoey] is that it has not successfully addressed problems in greater than two dimensions. Such a study would have important practical and theoretical benefits -- it would shed light on linear programming, multidimensional data analysis, geometric optimization, and retrieval on multiple keys, as well as provide a link between complexity and dimensionality. Some preliminary results are known. [Preparata and Hong] show that the convex hull of N points can be found in O(N log N) time in three dimensions but that $O(N^2)$ is a lower bound in any higher dimension. The maxima of a set of vectors can be found in $O(N \log^{k-2} N)$ time in dimension k [Kung et al.] and we conjecture that this time suffices to identify the extreme points of a k-dimensional set.

The most basic questions, however, have not been studied. These include finding nearest and farthest points, determining separability of point sets and other elementary properties. We intend in this paper to begin a systematic investigation of higher-dimensional geometry and its relation to complexity.

Closest-Point Problems

In this section we will investigate a number of problems dealing with the proximity of N points in Euclidean k-space. The most primitive closest-point problem is that of finding the two nearest of the N points; we will let P(N,k) denote the worst-case time of the best possible algorithm (the minimax complexity) for solving the closest-pair problem. The fixed radius-near-neighbor problem asks for all pairs of points within some fixed distance δ of one another. A special case of this problem arises when it is known that no ball of radius δ in the space contains more than some constant $c \geq 1$ number of points (this condition is known as sparsity); call the minimax complexity of the sparse fixed-radius-near-neighbor problem $S(N,k)$. The all-closest-points problem asks for the nearest neighbor of each of the N points; we will use $A(N,k)$ for its minimax complexity. We will assume the RAM/RASP model of computation in measuring the complexity of these problems [Aho, et al.].

All of these problems can be solved in a simple-minded way by investigating each of the $\binom{N}{2}$ pairs of points; the complexity of such an algorithm would be $O(kN^2)$. Algorithms that make use of special properties of the plane have shown that $P(N,2) = A(N,2) = O(N \log N)$ [Shamos and Hoey], but these algorithms do not generalize to k-space. We will present a divide-and-conquer algorithm for the closest-pair problem in the plane, generalize it to k-space, and extend the method to other closest-point problems. The divide-and-conquer method used to solve the problems is interesting in that the recursion operates by solving two problems on N/2 points in k-space, then projecting the remainder of the problem into a lower dimension. In this aspect it is similar to the algorithm for finding the maxima of a set of vectors given in [Kung, et al.].

We now give a class of divide-and-conquer algorithms for closest-point problems. We will proceed by presenting the basic divide-and-conquer algorithm for the closest-pair problem in the plane, generalizing it, then speeding up the generalizations. The algorithms, with the analyses of their worst-case running times, will be presented as proofs of theorems bounding the complexity of the problems. Because of its tractability we will work in the L_∞ metric to measure the distance between two points (the maximum coordinate metric), but use of any other L_p metric would change the time complexity of our algorithms only by some multiplicative constant. The L_∞ distance between two points (x_i, y_i) and (x_j, y_j) (in rectangular coordinates) is given by $d_\infty(i,j) = \max(|x_i - x_j|, |y_i - y_j|)$.

<u>Theorem 1</u>. $P(N,2) \leq O(N \log N)$.

<u>Proof</u>: To solve the problem for a collection of N points, divide the points by a vertical line L such that N/2 points lie on either side of L (Figure 1) and let A denote the leftmost N/2 points and B the rightmost. Recursively use this algorithm to find δ_A, the distance between the closest points of A, and similarly for δ_B. We will show how to obtain δ_{AB}, the distance between the two closest points of $A \cup B$, in $O(N)$ time. Let $\delta = \min(\delta_A, \delta_B)$. Figure 1 shows lines parallel to L on either side of it at distance δ. The area between these lines is a vertical slab of width 2δ.

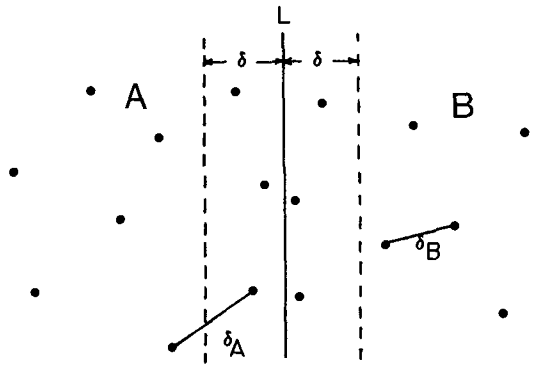

Figure 1.

If there is any pair in $A \cup B$ closer than δ apart, both points of the pair must lie in the slab. Now consider any square of side 2δ whose center lies on L (Figure 2). Since the minimum separation of points in the square on either side of L is δ, the square can contain at most a constant number c of points ($c = 12$ in this metric).

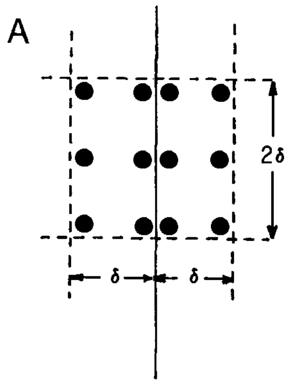

Figure 2.

Project all the points lying in the slab onto L. Within any interval of length 2δ on L there are at most 12 points. Since projection cannot increase the distance between two points, we can find all pairs of points within δ of each other in the slab by investigating all pairs of points within δ of each other in the projection. If there is a pair closer than δ apart in $A \cup B$ it will be found at this time. By the above observation, there are at most 12N pairs within δ in the projection, so if the points were presorted by y-coordinate, the checking could be done in linear time. Thus the recurrence relation describing the algorithm is $T(N) = 2T(N/2) + O(N)$, giving an $O(N \log N)$ algorithm. □

This procedure was discovered by [Strong] and can be generalized to any dimension.

Let us take careful note of the strategy behind this algorithm. The original problem dealing with N points in the plane is solved by solving two problems on N/2 points in the plane, then "patching up" the tentative solution by reducing the problem to one on N points in one dimension _in which sparsity is guaranteed_. We gain two important insights from this analysis: First, that divide and conquer can be used in multidimensional spaces, and second, that sparsity, though not present in the original set of points, can be induced in a subproblem. We will see later exactly how to induce sparsity; let us now examine the sparse problem in detail.

Theorem 2. $S(N,2) \leq O(N \log N)$.

Proof: We describe a divide-and-conquer algorithm similar to that given in the proof of Theorem 1. As before, we divide the points by a vertical line L into two sets A and B, each of size N/2 (this can be done in linear time after presorting or by a linear median algorithm [Blum et al.]). We then use this algorithm recursively to enumerate all pairs within distance δ of one another in A and similarly for B. We must now enumerate all pairs within distance δ of one another with one point in A and the other in B. To do this we consider all points in the slab given by the area within δ of L (note here that δ is fixed at the time the algorithm starts execution). Instead of having to prove sparsity as we did before (we showed $c = 12$), we know from the problem statement that no square of side 2δ in the plane has more than some constant c points (since a square of side 2δ is a δ-ball in the L_∞ metric). We now project all the points in the slab onto L and examine all pairs within δ of each other on L for the property of being within δ

in the plane. As before, if the points were presorted by y-coordinate, then the checking could be accomplished in linear time, giving again the recurrence $T(N) = 2T(N/2) + O(N)$, for an $O(N \log N)$ algorithm. $\square$

In the above algorithm we reduce the sparse problem in the plane to two sparse problems on $N/2$ points in the plane and one sparse problem on N points on the line. This strategy can immediately be generalized to collections of points in k-space, giving rise to the following theorem:

<u>Theorem 3</u>. $S(N,k) \le O(N \log^{k-1} N)$ for $k \ge 2$.

<u>Proof</u>: We will once again use a divide-and-conquer algorithm. We partition the points into two collections A and B, each having $N/2$ points. We induce this partition by P, a k-1 dimensional hyperplane perpendicular to the x-axis (for $k = 2$, the "hyperplane" was the 1-dimensional line L). We recursively enumerate all fixed-radius near-neighbors in A and similarly for B. We now must enumerate all fixed-radius near-neighbor pairs with one point in A and one in B. To accomplish this we project all points within δ of P onto P. Notice that this new set of points is still sparse with the <u>same</u> sparsity constant c. We now recursively use this algorithm on the (up to) N points in the (k-1)-dimensional problem. Because of the projection, not all the pairs within δ on P will be within δ in the original space, so an extra check will have to be made to ensure their proximity in the original space before they are reported (as well as ensuring that they are on different sides of P). This algorithm yields the recurrence relation $S(N,k) = 2S(N/2,k) + S(N,k-1) + O(N)$ (where the $O(N)$ is the overhead work done in dividing the points into the two collections A and B). Using from Theorem 2 the fact that $S(N,2) \le O(N \log N)$, we see by induction on k that the worst-case running time of the algorithm is bounded above by $O(N \log^{k-1} N)$. $\square$

In Theorem 3 the time-dependence on N seems to increase with k, a depressing but not a surprising prospect. If we examine the algorithm in closer detail, however, we see that the exponential factor in the logarithmic term comes from our willingness to solve a problem of size N in k-1 dimensions. If, however, we can bound the subproblem size by some function $f(N)$ such that $f(N) \log f(N) \le O(N)$, then we can easily prove by induction on k that $S(N,k) \le O(N \log N)$. (The recurrence used in the proof of Theorem 3 becomes

$$\begin{aligned} S(N,k) &= 2S(N/2,k) + O(N) + S(f(N),k-1) \\ &= 2S(N/2,k) + O(N) \end{aligned}$$

whose solution is $S(N,k) = O(N \log N)$.) Our strategy in bounding the problem size of the (k-1)-dimensional problem will involve a more intelligent choice of the cut-plane P used to partition the collection into the two subsets A and B. Specifically, we must choose a cut-plane that ensures two things: First, neither A nor B can contain "too many" points -- the subproblems in k-space must maintain <u>balance</u>. Second, there must be no more than $O(N/\log N)$ points within distance δ of P (this condition bounds the size of the subproblem to be solved in (k-1)-space). We proceed now by showing the existence of cut-planes with the above properties, first in 2-space and then in general k-space. We will then show how an algorithm can use these desirable cut-planes to reduce $S(N,k)$.

Theorem 4. (Existence of a cut-line in the plane.) Given a sparse collection of N points in the plane, there exists a cut line L perpendicular to one of the original coordinate axes with the following properties: (1) Both of the subcollections A and B induced by L contain at least N/8 of the points. (2) There are at most $2cN^{1/2}$ points within distance δ of L.

Proof: By contradiction: We demonstrate that a set without the properties described is at once very dense and very sparse. Consider the points sorted in increasing order by x-coordinate. Let us now restrict our discussion to only the middle 3N/4 points in the x-dimension (finding a cut-plane between two of the points in the restricted set will ensure condition 1 of the theorem). Assuming the opposite implies that every collection of $2cN^{1/2}$ points contiguous in the x-dimension projects onto a segment of the x-axis less than 2δ in length (if a given collection projected onto a wider interval, its center could be used to define a cut-line L with the desired properties). The situation that we have described is depicted in Figure 3. The regions R and L contain the right-most and leftmost N/8 points, respectively. The region C is that to which we have restricted our discussion; it contains 3N/4 points. The region T contains a set of $2cN^{1/2}$ points that are contiguous in the x dimension; we know that the width of T is no greater than 2δ. Since T is any collection of $2cN^{1/2}$ points in C, we can now bound the width of C by observing that C is composed of $(3N/4)/(2cN^{1/2})$ collections of $2cN^{1/2}$ points, and each of these collections is of width less than or equal to 2δ, so we can bound the width of C by

$$\text{width}(C) \le \frac{(3N/4)}{2cN^{1/2}} 2\delta = \frac{3N^{1/2}\delta}{4c} .$$

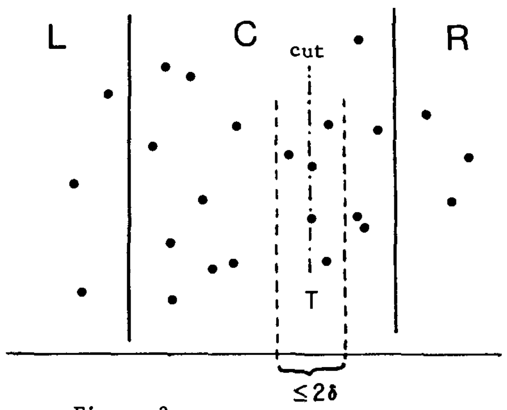

Figure 3.

We have thus far considered only C_x, the center points in the x-dimension; similar arguments could be made concerning C_y, the center 3N/4 points in the y-dimension. Let us now examine C_{xy}, the intersection of C_x and C_y. Since at most N/8 points are in each of L_x, L_y, R_x, and R_y, there must be at least N/2 points in C_{xy}. On the other hand, since the length of the sides of C_{xy} is bounded by $\frac{3N^{1/2}\delta}{4c}$, the total area of C_{xy} is bounded above by that length squared, or $\frac{9N\delta^2}{16c^2}$. By sparsity we know that the number of points in C_{xy} is therefore bounded above by $\frac{9\delta^2}{16c^2} \cdot \frac{c}{4\delta^2} = 9N/64c$. We have now shown that C_{xy} contains at least N/2 points (by hypothesis), but at most 9N/64c points for $c \ge 1$, which is a contradiction. Thus, the required cut-line exists. □

Theorem 5. (Existence of a cut-plane in k-space.) Given a sparse collection of N points in k-space, there exists a cut-plane P perpendicular to one of the original coordinate axes with the following properties: (1) Both of the subcollections A and B induced by P contain at least N/4k of the points. (2) There are at most $kcN^{1-1/k}$ points within distance δ of P.

Proof: The proof proceeds in the same manner as that of Theorem 4. Assuming the converse of the theorem leads us to the contradiction that the hypercube in k-space corresponding to C_{xy} contains at least N/2 points, but at most $\left[\frac{(1-\frac{1}{2k})}{kc}\right]^k \cdot cN$ points, which is less than N/2 for $k > 1$, $c \geq 1$. □

We will now show how the existence of cut-planes with the given properties allows us to speed up the sparse fixed-radius near-neighbor algorithm.

Theorem 6. $S(N,k) \leq O(N \log N)$.

Proof: We will modify the algorithm given in the proof of Theorem 3 to make use of the cut-planes of Theorem 5. The preprocessing of the points will include sorting them on each of the k coordinates; this can be done in O(k N log N) time. The recursive algorithm to enumerate all fixed-radius near-neighbor pairs will choose its cut-plane P by scanning each of the k sorted sequences until a point in some sequence is found that will define cut-plane -- Theorem 5 guarantees that such a cut plane exists. The scan involves investigating the middle $N(1 - 1/4k)$ points in each dimension until a collection of $kcN^{1-1/k}$ contiguous points is found which projects onto an interval of at most 2δ; this scan can be done in O(kN) time. We now enumerate all fixed-radius near-neighbor pairs in each of A and B, recursively. The cost of this step will be greatest when the problem sizes are most unbalanced which (by Condition 1 of Theorem 5) will be $S(N/4k,k) + S(N(1-1/4k),k)$. We now, as before, project all points within δ of P onto P. Notice (by Condition 2 of Theorem 5) that at most $kcN^{1-1/k}$ points will survive. Solving the subproblem in k-1 dimensions will therefore cost at most $S(kcN^{1-1/k},k-1)$. Thus the recurrence for the recursive algorithm is

$$S(N,k) \leq S(N/4k,k)+S(N(1-1/4k,k)+O(kN)+S(kcN^{1-1/k},k-1).$$

For any fixed k, the above recurrence yields $S(N,k) \leq O(N \log N)$. The proof of this, by induction on k, is based on the fact that if $S(N,k-1) \leq O(N \log N)$, then $S(kcN^{1-1/k},k-1) \leq O(N)$. Theorem 2 gives the basis for the induction. □

We end our discussion of the sparse fixed-radius near-neighbor problem by pointing out that sparsity is necessary in order to solve the problem in O(N log N) time. If the sparsity is not guaranteed, then it is possible to have all the points within some very small volume, giving $O(N^2)$ pairs of fixed-radius near-neighbors. Any correct algorithm to enumerate all such pairs would have time complexity greater than or equal to $O(N^2)$.

Now that we have some powerful tools with which to deal with the sparse fixed-radius near-neighbor problem, let us return to the closest-pair problem. The following theorem shows how one can employ the fast sparse fixed-radius near-neighbor algorithm to find closest pairs.

Theorem 7. $P(N,k) \leq O(N \log^2 N)$

Proof: We will generalize the algorithm given in the proof of Theorem 1 to k-space. To solve the closest-pair problem for a collection of N points, divide the points by a plane P perpendicular to the x-axis such that N/2 points lie on either side of P in regions A and B. Recursively use this algorithm to find δ_A, the distance between the two closest points in A, and similarly for δ_B. We will now show how to employ Theorem 6 to find δ_{AB}, the distance between the two closest points in $A \cup B$ in O(N log N) time. Let $\delta = \min(\delta_A, \delta_B)$. Just as in Theorem 1 we showed that no δ-ball in the plane can contain more than 12 points, a similar argument shows that no δ-ball in k-space can contain more than $4(3^{k-1})$ points. This same sparsity constant is preserved as we project all points within δ of P onto P. If there is any pair in the space closer than δ apart, that pair is now within δ on P. We can find all the fixed-radius near-neighbor pairs in P in time $O(N \log N)$, (by Theorem 6), and check these to determine which are closer than δ apart in k-space. The recurrence for this algorithm is $P(N,k) = 2P(N/2,k) + S(N,k-1) + O(N)$; using Theorem 6 we can show that $P(N,k) \le O(N \log^2 N)$. □

The logarithmic term in Theorem 7 is squared for the same reason that the logarithmic term in Theorem 3 is exponentiated -- we are willing to solve a subproblem in k-1 dimensions on as many as N points. We reduced the complexity of the sparse fixed-radius near-neighbor algorithm by an appropriate choice of cut plane; we can use a similar strategy for this problem. This will be a bit trickier, since sparsity is not guaranteed in the original set and we do not have a convenient value of δ as before.

Theorem 8. $P(N,k) \le O(N \log N)$.

Proof: Our proof will proceed in three stages. We will first show how the bound given in the theorem is achieved if an appropriate cut-plane can be found. We then demonstrate the existence of and a linear method for locating such a cut-plane in 2-space. The third stage of the proof demonstrates existence and shows how to find a suitable cut-plane in k-space.

We will modify the algorithm used in the proof of Theorem 7 to use a cut-plane P with the following properties (1) Both A and B (the sub-collections induced by P) contain at least N/4K of the points. (2) After $\delta = \min(\delta_A, \delta_B)$ has been found, there will be not more than $kcN^{1-1/k}$ points within δ of P. If these conditions are ensured, then the recurrence used in the proof of Theorem 7 becomes

$$\begin{aligned} P(N,k) &\le P(N/4k,k)+P(N(1-1/4k),k)+S(kcN^{1-1/k},k-1) \\ &\quad + O(N) \\ &= P(N/4k,k)+P(N(1-1/4k),k) + O(N) \end{aligned}$$

which gives $P(N,k) \le O(N \log N)$.

We will now give a method for finding a cut-plane with the above properties for the case of $k = 2$ (we will find a cut-line in the plane); we will later extend this method to arbitrary k. Let L be the line dividing the collection into parts with at least N/8 points which maximizes the length m needed to cover the projection of $2cN^{1/2}$ contiguous points onto one of the coordinate axes ($c = 12$ for $k = 2$). The following is a linear algorithm to find L, given that the lists are presorted: Consider every collection of $2cN^{1/2}$ points contiguous when projected on some axis. Let m be the interval between the extreme points in the projection. The cut line L associated with the collection is defined by the arithmetic mean of the extreme points. This situation is depicted in Figure 4. The area labelled T contains a collection of $2cN^{1/2}$ points contiguous in the x-dimension; the cut-line associated with T is labelled L. The linear scan to determine the maximum value of m starts in the second octile of the list and proceeds by keeping two pointers to points $2cN^{1/2}$ elements apart on the sorted list. The distance between these two points is noted and the interval m so observed is retained. The cut-line corresponding to largest such m is used to divide the problem and the algorithm proceeds as before, solving the two subproblems recursively. To show that there are at most $2cN^{1/2}$ points within δ of L it suffices to show that $m \geq 2\delta$ (for we know that there are only $2cN^{1/2}$ points within m/2 of L). If we assume that $m < 2\delta$, then we know that any projection of $2cN^{1/2}$ points projects onto an interval of length at most 2δ; otherwise the value of m we found would not be the maximum. But this condition on any collection of $2cN^{1/2}$ points which project onto a 2δ interval on one of the coordinate axes is what leads to a contradiction in our proof of Theorem 4. A similar argument at this point can lead us to the same contradiction here. We have thus shown that $m > 2\delta$, and we can find an appropriate cut-line for the closest point problem in the plane.

We can generalize this argument to k-space in the same way in which we generalized Theorem 4 to yield Theorem 5. The linear scan will bound the problem size by N/4k, and choose the cut plane P maximizing the distance m needed to cover $kcN^{1-1/k}$ contiguous points (where $c = 4(3^{k-1})$). It is easy to show that after δ is found, $m \geq 2\delta$, and thus the size of the (k-1)-dimensional subproblem is bounded by $kcN^{1-1/k}$. We have thus justified the recurrence given in the first stage of the proof and have the theorem as follows.

Our progress from Theorem 1 to Theorem 8 reveals a good deal about multidimensional algorithms. We have seen how divide-and-conquer can be employed in multidimensional spaces and have developed an interesting class of "doubly recursive" algorithms. We have demonstrated a relationship between the

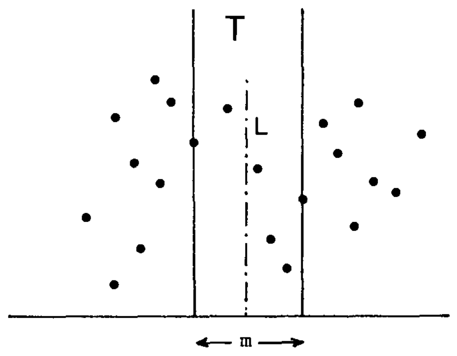

Figure 4.

closest-point problem and the sparse fixed-radius near-neighbor problem. Sparsity can be induced in problems for which it is not an input condition by judicious recursive subdivision. We have seen the merit of choosing cut planes wisely to limit the size of the subproblems to be solved in a space of reduced dimensionality. We might wonder if there exist more techniques that we might use to further reduce the bound on P(N,k). The following theorem tells us that we cannot reduce the bound; the algorithm given in the proof of Theorem 8 is optimal with respect to its dependence on N.

Theorem 9. $P(N,k) = O(N \log N)$ and this is optimal.

Proof: That $O(N \log N)$ is a lower bound on the time required to determine the two closest of N points in dimension one or higher is Theorem 1 of [Shamos and Hoey]. The proof uses the fact that $O(N \log N)$ is a lower bound on the element-uniqueness problem even if comparisons among linear functions of the input are allowed. We can use the closest-pair algorithm in k-space to test for element uniqueness by embedding the numbers to be tested for uniqueness on a line in k-space and then finding the closest pair in that space. The elements are unique if and only if the separation of closest pair is non-zero. □

Corollary 9.1. O(N log N) is a lower bound on the time necessary to find a minimum spanning tree on N points in any dimension.

Our analyses of both the closest pair problem and sparse fixed radius near neighbor problem have thus far centered about the total number of RAM/RASP operations as the measure of complexity. We gain an interesting insight into the problem, however, if we instead count the number of interpoint distance calculations performed during the execution of the algorithms. We will let $P_C(N,k)$ and $S_C(N,k)$ denote the minimax number of pointwise comparisons made by the closest-point and sparse fixed-radius near-neighbor algorithms, respectively. We then obtain the following unexpected result.

Theorem 10. $P_C(N,k) \le O(N)$ and $S_C(N,k) \le O(N)$.

Proof: The proof of this theorem proceeds by induction on k. With presorting it is obvious that $P_C(N,1) \le O(N)$ and $S_C(N,1) \le O(N)$. If we have established that $S_C(N,k-1) \le O(N)$, then the recurrence from the proof of Theorem 6, modified to count the number of pointwise comparisons, becomes

$$\begin{aligned} S_C(N,k) &\le S_C(N/4k,k)+S_C(N(1-1/4k),k)+S_C(kcN^{1-1/k},k-1) \\ &= S_C(N/4k,k)+S_C(N(1-1/4k,k)+O(N^{1-1/k}) \end{aligned}$$

which gives $S_C(N,k) = O(N)$. A similar argument shows that $P_C(N,k) = O(N)$. □

Though this theorem is of little use in improving the running time of the algorithm, it is interesting to note that most of the work of the algorithm is going into the presorting and the bookkeeping involved in recursion (along with choosing cut planes). Empirical tests have shown that the number of distance calculations used in solving the closest point problem in the plane is often strictly less than N.

All Closest Points and Minimum Spanning Trees

Theorem 9 says that we must spend O(N log N) time, in the worst case, in order to find the two closest points. Fortunately, a great deal more information can be obtained with very little additional work.

Theorem 11. $A(N,2) = O(N \log N)$. That is, for N points in the plane, the nearest neighbor of each can be found in O(N log N) time.

Proof: This result was obtained in [Shamos and Hoey] by Voronoi techniques which, although applicable in all dimensions, have only lead to fast algorithms in the plane. We give here a different method for two dimensions which immediately generalizes to k dimensions.

Partition the set of points via a vertical line L into subsets A and B, as in Theorem 1. Solve A(N/2,2) twice recursively, so at this stage we know, for each point in A its nearest neighbor in A and for each point in B its nearest neighbor in B. We must now find every point that is closer to some point in the other set than to any point in its own set. Let r(p) be the distance from p to its nearest neighbor in its own subset. By the recursive step, all the r(p) are known. For every point p, consider the r(p)-ball centered at p. In how many of these spheres centered in A does a given point ℓ of L lie? Figure 5 shows that this number is four in two dimensions under the L_2 (Euclidean) metric. (This follows from the maximum density of a point packing.) In any dimension k and L_p metric the maximum number of balls covering ℓ is some constant c(k,p). Project every point a of A onto L. Let $\ell(a)$ be the image of a on L.

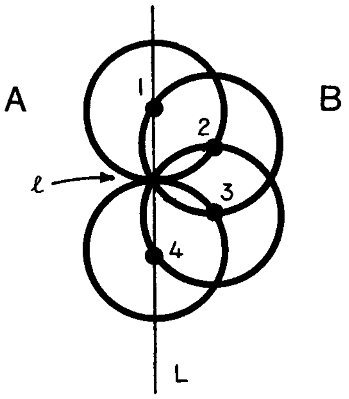

Figure 5.

In order to find the nearest neighbor of a in B, it is only necessary to examine those b whose r(b)-spheres contain $\ell(a)$. This means that for each of the N points of $A \cup B$, only a constant number other points need be examined. If the points are presorted by y-coordinate, all of the checking can be accomplished in linear time and $A(N,2) = 2A(N/2,2) + O(N) = O(N \log N)$. □

Corollary 11.1. $A(N,k) \le O(N \log^{k-1} N)$

Proof: Analogous to the proof of Theorem 3. □

A solution to the all nearest neighbors problem defines a graph in which points i and j are joined by a straight line iff i is a nearest neighbor of j or vice-versa. (If i has more than one nearest neighbor, only one is considered and without loss of generality the resulting graph is acyclic.) The graph is a minimal spanning forest, so A(N,k) solves the minimal spanning forest problem. A minimum spanning tree results if the all-nearest-neighbors graph is connected, but this normally does not happen fortuitously. Let T(N,k) be the time necessary to find a minimum spanning tree on N points in k dimensions under any L_p metric.

Theorem 12. $T(N,2) = O(N \log N)$

Proof: [Shamos and Hoey] □

Theorem 13. $T(N,k) \ge A(N,k)$.

Proof: Immediate. It follows from the fact that the nearest-neighbor graph is a subgraph of the minimum spanning tree. □

Conjectures and Unsolved Problems

The structure of the closest-point algorithms in high dimensions leads us to conjecture that $T(N,k) = O(N \log N)$. Some credence is given to this prediction by the studies of [Bentley and Friedman] who show that a "nearly minimal" spanning tree can be constructed in any constant dimension in $O(N \log N)$ time.

A serious difficulty with these "fast" algorithms is that their complexity is exponential in k, the dimension. This is particularly annoying because naive algorithms for the nearest neighbor and minimum spanning tree problem run in $O(kN^2)$ time, linear in dimension. We know of no way to avoid the exponential explosion in dimension that occurs when we attempt to develop an algorithm that is of optimal order in N. The issue comes to a head in the problem of finding the two closest of N points in N dimensions. The naive algorithm requires $O(N^3)$ time and there are N^2 independent inputs. We conjecture that an $O(N^2 \log N)$ algorithm exists and is optimal.

The connection between closest and farthest points is curious. If F(N,k) denotes the time required to find the two farthest of N points in k dimensions, we have $O(N) = F(N,1) < P(N,1) = O(N \log N)$. That is, in one dimension, finding farthest points is easier than finding closest points. In two dimensions, they have the same complexity [Shamos and Hoey]. However, no divide-and-conquer algorithm is known for the farthest-points problem that generalizes to higher dimensions. There is some indication that $F(N,3) = O(N \log N)$ (see [Preparata and Hong]) but it has not even been shown that $F(N,k) < O(N^2)$ for $k > 3$.

It is natural when considering closest and farthest pairs to ask how quickly the mth closest pair can be found in k dimensions. For $m = N/2$ this is the median interpoint distance, whose minimax time complexity we denote by $M(N,k)$. The problem can be solved in one dimension in $O(N \log N)$ time but $M(N,k)$ is not known to be less than $O(N^2)$ for $k > 1$.

Summary

We have tried to broaden the scope of computational geometry by extending its reach to an arbitrary number of dimensions. Using a method of recursion in both problem size and dimension has enabled us to improve radically the time bounds on multidimensional problems. Unfortunately, at each recursion step we are only able to reduce the dimension by one, while the number of points is reduced by half. This means that the algorithms are much more sensitive to an increase in the number of dimensions than they are to an increase in the number of data points. This effect appears to be a general feature of geometric algorithms.

Acknowledgments

J. L. B. would like to thank Andrew and Frances Yao and Charles T. Zahn for helpful discussions. This work was performed while he was a National Science Foundation Fellow and was visiting at the Stanford Linear Accelerator Center and at Carnegie-Mellon University.

M. I. S. is indebted to H. R. Strong for an elegant idea which gave rise to all of the algorithms presented here.

References

Aho, A. V., Hopcroft, J. and Ullman, J. D. The Design and Analysis of Computer Algorithms. Addison-Wesley (1973).

Bentley, J. L. and Friedman, J. Fast Algorithms for Constructing Minimal Spanning Trees in Coordinate Spaces. SLAC Technical Report.

Blum, M., et al. "Time Bounds for Selection." JCSS 7 (1973), 448-461.

Kung, H. T., Luccio, F. and Preparata, F. P. "On Finding the Maxima of a Set of Vectors," JACM 22 (1975), 469-476.

Preparata, F. P. and Hong, S. J. Convex Hulls of Finite Planar and Spatial Sets of Points. Report R-682, Coordinated Science Laboratory, University of Illinois (April, 1975).

Shamos, M. I. "Geometric Complexity." Proc. 7th Annual Symposium in Theory of Computing. May, 1975.

Shamos, M. I. "Geometry and Statistics: Problems at the Interface." Proc. Symposium on Algorithms and Complexity, Carnegie-Mellon University. April, 1975.

Shamos, M. I. "Problems in Computational Geometry." Unpublished manuscript.

Shamos, M. I. and Hoey, D. J. "Closest-Point Problems." Proc. 16th Annual Symposium on Foundations of Computer Science. October, 1975.

Strong, H. R. Private communication.