[8] K. J. Supowit, “The relative neighborhood graph with an application to minimum spanning trees,” Tech. Rept., Department of Computer Science, University of Illinois, Urbana-Champaign, August 1980, also to appear in J.A.C.M., 1983.

[9] D. W. Matula and R. R. Sokal, “Properties of Gabriel graphs relevant to geographic variation research and the clustering of points in the plane,” *Geographical Analysis*, Vol. 12, July 1980, pp. 205-222.

[10] D. T. Lee and B. J. Schacter, “Two algorithms for constructing a Delaunay triangulation,” *International Journal of Computer and Information Sciences*, Vol. 9, No. 3, 1980, pp. 219-242.

[11] R. N. Horspool, “Constructing the Voronoi diagram in the plane”, Tech. Report No. SOCS 79.12, School of Computer Science, McGill University, July 1979.

[12] M. McKenna and G.T. Toussaint, “Finding the minimum vertex distance between two disjoint convex polygons in linear time”, Tech. Report No. SOCS-83.6, School of Computer Science, McGill University, April 1983.

[13] G. T. Toussaint, “An optimal algorithm for computing the minimum vertex distance between two crossing convex polygons”, Technical Report No. SOCS-83.7, School of Computer Science, McGill Univ., May 1983.

[14] A. Yao, “On constructing minimum spanning trees in K-dimensional spaces and related problems”, *SIAM Journal on Computing*, Vol. 11, No. 4, November 1982, pp. 721-736.

Amdahl V-7 computer using a FORTRAN-G1 compiler. The Gabriel graph of the set $S_1 \cup S_2$ was constructed by first constructing the Voronoi diagram of $S_1 \cup S_2$ using the computer program developed by Horspool [11]. As the results show, the brute-force method is efficient for small values of *n*, but for other values of *n*, MIN is far superior to the brute-force method.

6. Concluding Remarks

In this paper we have shown that the minimum distance between two finite planar sets of points can be computed in O(*n* log *n*) worst-case running time and that this is optimal to within a constant factor. Furthermore when the sets form a convex polygon this complexity can be reduced to O(*n*). We have also discussed the relation between the set-distance problem and the closest-pair problem. A generalization of the set-distance problem is what might be termed the *k-color distance* problem: given a planar set of *n* points colored arbitrarily with k colors find the closest pair of points of different color. When $k = 1$ this problem reduces to the closest-pair problem. When $k = 2$ it reduces to the minimum set-distance problem. Note that when $k = 3$, the closest pair of points of two *specified* colors need not correspond to an edge of the MST of *S*, where $S = S_1 \cup S_2 \cup S_3$.

A more difficult problem for the case of convex polygons calls for finding the closest pair of vertices p_i, q_j between two convex polygons $P = (p_1, p_2, ..., p_n)$ and $Q = (q_1, q_2, ..., q_n)$, where, $p_i \in P$ and $q_j \in Q$. Recently, O(*n*) algorithms have also been discovered for this problem [12], [13].

One open problem concerns the case of higher dimensions. Can we obtain O(*n* log *n*) algorithms for finding $d_{min}(S_1, S_2)$ in any fixed dimension $k > 2$? That we are not doomed to O(n^2) algorithms has been recently shown by Yao [14] for the restricted case of *widely separated sets* S_1 and S_2, i.e., sets such that $d_{min}(S_1, S_2) > n(diam(S_1) + diam(S_2))$, where *diam* denotes diameter. Under this assumption Yao showed that $d_{min}(S_1, S_2)$ may be computed in O((*n* log *n*)$^{1.8}$) time when k=3.

7. References

[1] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, John Wiley, New York, 1973, p. 235.

[2] D. Avis, "Lower bounds for geometric problems," *Proc. Allerton Conference*, Urbana, Illinois, October 1980.

[3] M. I. Shamos, "Computational geometry," Ph.D. thesis, Yale University, 1978.

[4] G. T. Toussaint, "Pattern recognition and geometrical complexity," *Proc. Fifth International Conference on Pattern Recognition*, Miami Beach, December 1980, pp. 1324-1347.

[5] M. I. Shamos, "Geometric complexity," *Proc. 7th ACM Symposium on the Theory of Computing*, May 1975, pp. 224-233.

[6] D. T. Lee and F. P. Preparata, "The all nearest-neighborhood problem for convex polygons," Information Processing Letters, Vol. 7, June 1978, pp. 189-192.

[7] G. T. Toussaint, "The relative neighborhood graph of a finite planar set," *Pattern Recognition*, Vol. 12, 1980, pp. 261-268.

5. Experimental Results

In this section, we describe the implementation of one of the above algorithms and compare it to the brute-force method. The implementation uses the Gabriel graph because this graph is easy to obtain from the Voronoi diagram. An edge in the Delaunay triangulation is an edge in the Gabriel graph if it intersects its corresponding Voronoi edge [9].

ALGORITHM MIN

begin

Step 1: Compute the Voronoi diagram of $S = S_1 \cup S_2$.

Step 2: From the dual of the Voronoi diagram, obtain the Gabriel graph of S.

Step 3: From among the edges of the Gabriel graph, determine the set of pairs of points such that one is from S_1 and the other from S_2.

Step 4: Compute the distance between all pairs of points in Step 3.

Step 5: Determine the minimum of the distances computed in Step 4.

end

To evaluate the performance of algorithm MIN against the brute-force method, a Monte Carlo simulation was carried out. Points of both sets S_1 and S_2 were such that each set was distributed uniformly in a unit square. The size of each set considered was n = 100, 500, 1000 or 2500. Each case was repeated a number of times and the running times for both the brute-force method and the MIN are shown in Table 1. These running times were obtained by carrying out a simulation in an

ALGORITHM	n = 100 (75)		n = 500 (75)		n = 1000 (25)		n = 2500 (15)	
	Time (secs)	Std. dev.	Time (secs)	Std. dev.	Time (secs)	Std. dev.	Time (secs)	Std. dev.
BRUTE-FORCE	0.017	0.0000	0.618	0.0041	2.530	0.0082	16.333	0.0102
MIN	0.083	0.0019	0.523	0.0039	1.129	0.0132	2.621	0.0531

Table 1: Running times to compute the minimum distance between two separable sets each containing n points. The points of each set were such that each set was distributed uniformly in a unit square. The numbers inside the parentheses indicate the number of times the experiment was repeated.

Proof: Let $p \in S_1$ and $q \in S_2$ determine the minimum distance between S_1 and S_2 and assume edge $\overline{pq}$ is not contained in MST(S). Then p and q are connected in the MST of S by a path through at least one other vertex of MST(S). Furthermore, on this path there exists an edge joining two vertices p', q' such that $p' \in S_1$ and $q' \in S_2$. Note that it is possible that $p' = p$ or $q' = q$ but not both. Since $\overline{p'q'}$ is contained in MST(S) it follows that $d(p',q') < d(p, q)$ which is a contradiction. If $\overline{pq}$ is contained in MST(S) it must be contained in any sub-graph of the MST. This proves the sufficiency for G*(S) to exhibit the *edge-inclusion* property. To see that it is also a necessary condition consider an MST in which k edges are removed leaving $k+1$ trees $T_1, T_2,..., T_{k+1}$. Let $T_1 \in S_1$ and $T_2,..., T_{k+1} \in S_2$. Then clearly $d_{min}(S_1, S_2)$ is realized by an edge which is not contained in $T_1 \cup T_2 \cup ... \cup T_{k+1}$.

Q.E.D.

The above theorem suggests algorithms for computing $d_{min}(S_1, S_2)$ and also characterizes the difference between the *set-distance* problem and the *closest-pair* problem. In the latter problem it is *not* necessary that G*(S) be a sub-graph of MST(S) for G*(S) to always contain the edge defining the closest pair.

3. Algorithms for Planar Sets of Points

There exist several graphs on S that contain O(n) edges, can be computed in O(n log n) worst-case running time and O(n) expected time under certain probability models, and exhibit the *edge-inclusion* property for $d_{min}(S_1, S_2)$. Among the more well known are the MST [3], the relative neighborhood graph (RNG) [7], [8], the Gabriel graph (GG) [9], and the Delaunay triangulation (DT) [10]. In fact the above graphs are related to each other as follows [4]:

$$\text{MST} \subseteq \text{RNG} \subseteq \text{GG} \subseteq \text{DT}.$$

Any of the above algorithms [3], [7]-[10], could be used to obtain an O(n log n) algorithm for computing $d_{min}(S_1, S_2)$. Once the graph for S is obtained it is a trivial matter to check its edges sequentially ignoring those containing vertices belonging to the same set and updating a tentative answer whenever a smaller set-distance is encountered. Since these graphs contain no more than $3n$-6 edges O(n) time suffices for this step. Furthermore, Avis [2] has shown that $\Omega(n \log n)$ is a lower bound for this problem and hence the above algorithms are optimal to within a constant factor. It should also be noted that the above algorithms could be used to compute the closest pair as well, although it is doubtful that in practice they would be faster than computing the closest pair directly via divide and conquer or by first computing the *nearest neighbor graph* [3].

4. An O(n) Algorithm for Convex Polygons

The set distance problem for convex polygons can be easily stated as the following coloring problem. Given a convex polygon with its vertices arbitrarily colored with two colors, find the closest pair of vertices of opposite colors. Close refers again to smallest euclidean distance. Supowit [8] has designed an O(n) worst-case algorithm for computing the RNG of a convex polygon. Once the RNG is obtained d_{min} can be computed in O(n) time leading to an O(n) worst-case running time algorithm for the set-distance problem for convex polygons.

where $m_1 = \frac{1}{n}\sum_{i=1}^{n} p_i$ and $m_2 = \frac{1}{n}\sum_{i=1}^{n} q_i$

Clearly $d_{mean}(S_1, S_2)$ can always be computed in O(n) time. Duda and Hart [1] point out that d_{mean} is computationally more attractive than d_{min} by claiming that d_{min} requires the computation of all n^2 distances, resulting in an O(n^2) algorithm.

In this paper we show that $d_{min}(S_1, S_2)$ can be computed in O(n log n) time in the worst-case. Since $\Omega(n \log n)$ is a lower bound for this problem [2], the algorithms given here are optimal to within a multiplicative constant. In addition the algorithms exhibit O(n) expected running time for a wide class of underlying distributions of the points. Thus, even in the worst case, it is not necessary to compute n^2 distances. Furthermore, when S is a convex polygon, the vertices of which are arbitrarily colored with two colors, it is shown that the closest pair of vertices of opposite colors can be found in O(n) time in the worst case. These results generalize existing results on the *closest-pair* problem.

2. Minimum Set Distance and the Closest Pair

The *closest-pair* problem consists of determining the pair of points in a set S that are closest together in the sense of minimizing the euclidean distance. This problem has received attention recently in the literature on computational geometry [3], [4]. Shamos [5] has shown that $\Omega(n \log n)$ is a lower bound for this problem, and in [3] presents a divide-and-conquer algorithm that runs in O(n log n) time, and is thus optimal. When the input is a convex n-vertex polygon rather than an arbitrary set of points the $\Omega(n \log n)$ lower bound does not hold. Lee and Preparata [6] show that this additional property is sufficient to obtain an O(n) algorithm. Actually Lee and Preparata [6] solve the more general problem of finding the nearest neighbor of every vertex (the *nearest neighbor graph* is obtained by joining with an edge every vertex to its nearest neighbor). One such pair will be the *closest pair*.

The *closest pair* problem appears similar in some sense to the *minimum set-distance* problem. That the two problems are not identical is obvious from the observation that the closest pair in S may consist of two points both of which belong to S_i, $i = 1, 2$. In fact the pair of points $p \in S_1$, $q \in S_2$ realizing the minimum set distance do not even correspond to an edge that is guaranteed to be contained in the *nearest neighbor graph* (NNG) of S, i.e., the NNG does not exhibit the *edge-inclusion* property for the minimum set-distance edge. Thus we cannot solve the set distance problem by first computing NNG(S) and selecting its shortest edge connecting a point in S_1 with another point in S_2.

On the other hand it is trivial that the complete graph on S, G(S) does exhibit the *edge-inclusion* property for d_{min}. However choosing d_{min} by examining all the edges of G(S) leads to an O(n^2) algorithm. Thus we are interested in sub-graphs of G(S), say G*(S), that can be computed in o(n^2) time, that have preferably O(n) edges, and that exhibit the *edge-inclusion* property for d_{min}. We show that a necessary and sufficient condition for G*(S) to exhibit the *edge-inclusion* property is that G*(S) be the minimal spanning tree (MST) of S.

Theorem 2.1: Given a set $S = S_1 \cup S_2$, the pair of points $p \in S_1$, $q \in S_2$ realizing the minimum distance, determine an edge in the MST of S

Optimal Algorithms for Computing the Minimum Distance Between Two Finite Planar Sets

Godfried T. Toussaint
and
Binay K. Bhattacharya

ABSTRACT

It is shown in this paper that the minimum distance between two finite planar sets if n points can be computed in O(n log n) worst-case running time and that this is optimal to within a constant factor. Furthermore, when the sets form a convex polygon this complexity can be reduced to O(n).

Index terms: minimum distance between sets, cluster analysis, pattern recognition, coloring problems, convex polygons, algorithms, computational geometry, complexity.

CR Categories: 3.63, 5.24, 5.25

1. Introduction

Let $S_1 = \{p_1, p_2, ..., p_n\}$ and $S_2 = \{q_1, q_2, ..., q_n\}$ be two planar sets of n points each and let $S = S_1 \cup S_2$. The sets need not have equal cardinality but such an assumption simplifies notation. A point p_i is given in terms of the cartesian coordinates x_i and y_i. The minimum distance between S_1 and S_2, denoted by $d_{min}(S_1, S_2)$, is defined as

$$d_{min}(S_1, S_2) = \min_{i,\, j} \{d(p_i, q_j)\},\ i, j = 1, 2, \ldots, n$$

where $d(p_i, q_j)$ is the euclidean distance between p_i and q_j.

Another frequently used distance between sets is given by:

$$d_{mean}(S_1, S_2) = d(m_1, m_2)$$