Algorithmica (1993) 9: 398–423

Algorithmica
© 1993 Springer-Verlag New York Inc.

The Searching over Separators Strategy To Solve Some NP-Hard Problems in Subexponential Time[1]

R. Z. Hwang,[2] R. C. Chang,[3,4] and R. C. T. Lee[2,4]

Abstract. In this paper we propose a new strategy for designing algorithms, called the searching over separators strategy. Suppose that we have a problem where the divide-and-conquer strategy can not be applied directly. Yet, also suppose that in an optimal solution to this problem, there exists a separator which divides the input points into two parts, A_d and C_d, in such a way that after solving these two subproblems with A_d and C_d as inputs, respectively, we can merge the respective subsolutions into an optimal solution. Let us further assume that this problem is an optimization problem. In this case our searching over separators strategy will use a separator generator to generate all possible separators. For each separator, the problem is solved by the divide-and-conquer strategy. If the separator generator is guaranteed to generate the desired separator existing in an optimal solution, our searching over separators strategy will always produce an optimal solution. The performance of our approach will critically depend upon the performance of the separator generator. It will perform well if the total number of separators generated is relatively small. We apply this approach to solve the discrete Euclidean P-median problem (DEPM), the discrete Euclidean P-center problem (DEPC), the Euclidean P-center problem (EPC), and the Euclidean traveling salesperson problem (ETSP). We propose $O(n^{O(\sqrt{P})})$ algorithms for the DEPM problem, the DEPC problem, and the EPC problem, and we propose an $O(n^{O(\sqrt{n})})$ algorithm for the ETSP problem, where n is the number of input points.

Key Words. Computational geometry, NP-hardness.

1. Introduction. The divide-and-conquer strategy is a well-known approach to designing efficient algorithms (Aho *et al.*, 1976; Preparata and Shamos, 1985; Horowitz and Sahni, 1978; Bentley, 1976, 1980). The basic idea of this approach is as follows:

(1) We divide the input data into two subsets A_d and C_d.
(2) We then recursively solve the two subproblems with A_d and C_d as input data, respectively.
(3) Finally, we find an efficient way to merge the solutions to the two subproblems into the solution to the original one.

A typical example is the closest-pair problem, which is defined as follows: given a set of n points in the plane, find the closest pair of these points. We can solve this problem by the divide-and-conquer approach. First, we find a median line

[1] This research work was partially supported by the National Science Council of the Republic of China under Grant NSC 79-0408-E007-04.
[2] Institute of Computer Science, National Tsing Hua University, Hsinchu, Taiwan, Republic of China.
[3] Institute of Computer Science, National Chiao Tung University, Hsinchu, Taiwan, Republic of China.
[4] Academia Sinica, Taipei, Taiwan, Republic of China.

Received August 4, 1990; revised March 15, 1991. Communicated by C. L. Liu.

which divides the data into two subsets of equal size. We then find the closest pairs in A_d and C_d, respectively. Let the distance of the closest pair in A_d (resp. C_d) be denoted as d_a (resp. d_b), and $d = \min(d_a, d_b)$. To find the closest pair of the entire set, we only have to examine the points in the strip which centers at the median line with $2 \cdot d$ as its width. For more details, consult Bentley (1976) and Preparata and Shamos (1985).

Many problems can be solved by the divide-and-conquer strategy (Horowitz and Sahni, 1978). It usually yields efficient polynomial algorithms. Unfortunately, not every problem can be solved efficiently by the divide-and-conquer approach. One of the reasons is that we cannot easily divide the input data into two unrelated subsets, such that the two subproblems with these two subsets as inputs can be solved independently and the solution later merged into an optimal solution.

In this paper we point out that there may exist cases characterized by the following properties:

(1) The problem is an NP-hard problem. Thus it is quite unlikely that it can be solved by the divide-and-conquer strategy directly.
(2) On the other hand, as far as an optimal solution S is concerned, there exists a separator, called B. The separator B divides the input data D into two parts A_d and C_d. Then the final solution can be derived by merging the optimal solutions to subproblems A_d and C_d.
(3) The separator B has some kind of characteristic. All possible separators with such properties can be generated efficiently.

If a problem has the above properties, then although it cannot be solved by the divide-and-conquer strategy, it can be solved by *the searching over separators strategy* which we now propose and explain in the following:

Let us further assume that our problem is an optimization problem, and we are looking for an optimal solution with the minimum cost. The searching over separators strategy works as follows:

```
The Searching over Separators Strategy
Input: A set of input data $D$.
Output: An optimal solution $S$ and its cost $C$.

Step 1. Let $C := \infty$.
Step 2. Use some procedure, called Procedure A, to generate all possible
        separators.
Step 3. For each possible separator B, do:
Step 4.     Use B to divide the input data $D$ into two subsets $A_d$ and $C_d$.
Step 5.     Recursively solve the subproblems with $A_d$ and $C_d$ as inputs,
            respectively. Let the solutions be $A_s$ and $C_s$, respectively.
Step 6.     Merge $A_s$ and $C_s$ into $S'$. Let $C'$ be the cost associated with $S'$.
Step 7.     If $C > C'$, then $S := S'$, $C := C'$.
Step 8. Return solution $S$ as an optimal solution and the optimal cost $C$.
```

Let us consider a case where we are given a set of n points and we are to select P points out of them to form an optimal solution. A straightforward approach is to examine every possible subset of P points. There are $\binom{n}{P}$ possible combinations. Therefore, the time complexity of this straightforward approach is at least

$$O\left(\binom{n}{P}\right) = O(n^P).$$

Now suppose that we apply the searching over separators strategy to solve this problem. Let $T(n, P)$ be the time complexity of solving this problem by using the searching over separators approach. Assume that the separator B has some kind of characteristic, such that, after dividing, the time complexity of each subproblem is bounded by $T(n, \alpha \cdot P)$ and the total number of possible separators and the time needed to generate these separators are both bounded by $O(n^{c \cdot \sqrt{P}})$, where $0 < \alpha < 1$ and c is some constant. Then we have the following formula:

$$\begin{aligned}
T(n, P) &\le O(n^{c \cdot \sqrt{P}}) \cdot 2 \cdot T(n, \alpha \cdot P) \\
&\le O(n^{c \cdot \sqrt{P}} \cdot 2 \cdot n^{c \cdot \sqrt{\alpha \cdot P}}) \cdot 2 \cdot T(n, \alpha^2 \cdot P) \\
&= O(n^{c \cdot \sqrt{P}(1 + \sqrt{\alpha})}) \cdot 2 \cdot T(n, \alpha^2 \cdot P) \\
&\le O(n^{c \cdot \sqrt{P}(1 + \sqrt{\alpha} + \sqrt{\alpha^2} + \cdots)}) \\
&\le O(n^{O(\sqrt{P})}).
\end{aligned}$$

Compared with the straightforward approach, the searching over separators strategy achieves a better performance.

In order for the above searching over separators strategy to work. Procedure A must be able to produce the particular separator B existing in an optimal solution as described above. If it does, the searching over separators strategy is guaranteed to have examined an optimal solution and correctly present it as a solution.

In this paper we show that the searching over separators strategy can be used to solve four geometry problems: the discrete Euclidean P-median problem (DEPM), the discrete Euclidean P-center problem (DEPC), the Euclidean P-center problem (EPC), and the Euclidean traveling salesperson problem (ETSP).

This paper is organized as follows: In Section 2 we solve the DEPM problem and the DEPC problems. In Section 3 we extend the method in Section 2 to solve the EPC problem. In Section 4 we solve the ETSP problem. Finally, the concluding remarks are stated in Section 5.

2. The Algorithm To Solve the Discrete Euclidean P-Median Problem and the Discrete Euclidean P-Center Problem

2.1. Preliminaries. The EPM problem is defined as follows: given n demand points on the plane, the EPM problem is to select P locations as supply points, such that the sum of distances from all demand points to their respective nearest supply points is minimized. The DEPM problem is its related problem, in which the supply points must be selected from the set of the given demand points. There are many real-world applications of the DEPM problem. One of these applications is to choose P locations to build warehouses, such that the sum of distances from all stores to their respective nearest selected warehouses is minimized.

Formally we can formulate the DEPM problem as follows: given a set $D = \{d_1, d_2, \ldots, d_n\}$ of demand points, find a set S of P supply points from the set of demand points, to minimize the following cost function:

$$\sum_{d_i \in D} \left\{ \min_{s_j \in S} \{ l(d_i, s_j) \} \right\}, \qquad \text{where } l(d_i, s_j) \text{ is the Euclidean distance between } d_i \text{ and } s_j.$$

Megiddo and Supowit (1984) proved that the EPM problem is an NP-hard problem. Papadimitriou (1981) proved the NP-hardness of the DEPM problem.

The DEPM problem can be solved by enumerating all possible combinations of P points as the supply points and selecting the set of points which minimizes the total sum of distances from the n demand points to their respective nearest supply points. This approach takes $O(P \cdot n^{P+1})$ time (Papadimitriou, 1981). In this paper we solve the DEPM problem in $O(n^{O(\sqrt{P})})$ time.

2.2. The Generalized Discrete Euclidean P-Median Problem. For reasons which will become clear later, we try to solve a generalized version of the DEPM problem instead of the original problem. Let us now modify the original DEPM problem into *the generalized discrete Euclidean P-median problem (GDEPM)*. The GDEPM problem is defined as follows: given a set D of n demand points, a set $\beta \subset D$ of fixed supply points, and the number P, we have to select a set S of P supply points from D, where β and S are disjoint, such that

$$\sum_{d_i \in D} \left\{ \min_{s_j \in S \cup \beta} \{ l(d_i, s_j) \} \right\} \qquad \text{is minimized.}$$

To distinguish the different problems, we use the GDEPM-(P, D, β) problem to denote the GDEPM problem with P, D, β as inputs.

We can immediately see that the original DEPM problem is a special case of the GDEPM problem in which β is an empty set. Note that we have to distinguish two kinds of supply points. Thus, throughout the rest of this paper, we use β to denote the set of the fixed supply points, and use S to denote the set of the unfixed supply points.

Essentially, we show that, for an optimal solution S to the GDEPM problem, there exists a cycle, named *the B-cycle*. Let $B_s \subset S$ denote the set of unfixed supply points in the B-cycle. The B-cycle has the following properties:

(1) $|B_s|$ is no more than $\sqrt{8 \cdot (P + 3)}$. (Note that $|B_s|$ denotes the number of points in the set $|B_s|$.)
(2) The B-cycle divides the other unfixed supply points in S into the interior part A_s and the exterior part C_s, where $|A_s|, |C_s| \le 2P/3$.

(3) The B-cycle divides D into interior part A_d and exterior part C_d. For each demand point in A_d (resp. C_d), its nearest supply point is in A_s (resp. C_s) or $B_s \cup \beta$.

The third property implies that we can divide the original problem into two subproblems with $(|A_s|, A_d, B_s \cup \beta)$ and $(|C_s|, C_d, B_s \cup \beta)$ as inputs. We can see that the solution to the original problem can be obtained by merging the solutions to the two subproblems. The first property guarantees that the number of possible separators can be bounded by a tolerable value. The second property guarantees that the number unfixed supply points in each subproblem is at most $2P/3$.

To show the existence of the B-cycle, we first note that we can construct a Delaunay triangulation (Preparata and Shamos, 1985) out of the unfixed supply points of an optimal solution with some special arrangement, such that this Delaunay triangulation is a maximal planar graph in which very face is of size 3 (Nishizeki and Chiba, 1988). Therefore we can use the simple cycle separator theorem proved by Miller (1986), which was in turn based upon the planar separator theorem proved by Lipton and Tarjan (Mehlhorn, 1984; Lipton and Tarjan, 1979). For a comprehensive discussion of this topic, consult Nishizeki and Chiba (1988).

Miller (1986) assumed that we were given a planar graph G with nonnegative weights assigned to vertices, faces, and edges which sum to 1. For our case, we simply assume that the weights of faces and edges are all zeros. In other words, weights are assigned only to vertices. For a simple cycle B of G, the size of this cycle is the number of the vertices on B. Note that a simple cycle of G will always divide G into two parts, the interior A and the exterior C. The weight of the interior part (resp. the exterior part) is the sum of weights of vertices in the interior part (resp. the exterior part). Figure 2.1 shows a planar graph and the thick edges form a simple cycle. For this case, the size of the cycle, the weight of its interior, and the weight of its exterior are 7, 0.35, and 0.45, respectively.

The theorem proved by Miller (1986) is now stated as follows. (Note that it is slightly different from that original one stated by Miller (1986), because we do not assign weights to faces and edges.)

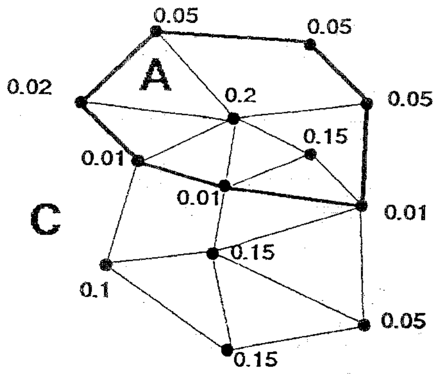

Fig. 2.1. A planar graph. The weight of $A = 0.35 < \frac{2}{3}$ and the weight of $C = 0.45 < \frac{2}{3}$.

THEOREM 2.1 (Miller, 1986). *If G is a 2-connected planar graph with all nonnegative weights assigned to vertices which sum to 1, there exists a simple cycle, called the simple cycle separator, of size at most $2\sqrt{2\lfloor d/2\rfloor k}$, dividing the graph into interior and exterior two parts, such that the sum of the weights in both parts is no more than two-thirds, where d is the maximum face size (the face size is the number of edges contained in the facial cycle) and k is the number of vertices in G.*

In our case we are interested in the maximal planar graph (Nishizeki and Chiba, 1988), where d is equal to 3. Hence the size of the separator is accordingly $\sqrt{8k}$.

Given a set S of P unfixed supply points, the Delaunay triangulation of S cannot be a maximal planar graph, because the outer face is not necessarily of size 3. There is a simple way to change the Delaunay triangulation graph into a maximal planar graph, by adding a set I of three extra points to form a triangle enclosing all points in S. Let $S' = S \cup I$. The Delaunay triangulation of S' is a maximal planar graph, because its outer face contains exactly three edges and other faces are triangulated. Hence we can apply Theorem 2.1 to the Delaunay triangulation of S'.

In our algorithm we treat the three points of the triangle as supply points, and they are not related to any demand point. We define the enclosing I as follows: the enclosing points are three points which form a triangular boundary enclosing all points, such that no demand point's distance to its closest supply point is longer than its distance to any of the enclosing points. We can see that it is trivial to find these enclosing points. We simply select them far enough from any of the demand points.

By applying Theorem 2.1 to S', we can see that there exists a simple cycle separator B with size less than $\sqrt{8(P+3)}$ which divides the Delaunay triangulation graph of S' into the interior and exterior parts, such that the number of unfixed supply points in each part is less than $2P/3$. This result is derived by assigning zero weights to the enclosing points and equal weights to the unfixed supply points.

Nevertheless, Miller's simple cycle separator theorem cannot guarantee that the sets of D and S divided by the simple cycle separator satisfy the third property of the B-cycle. This is due to the fact that we construct our Delaunay triangulation graph out of the unfixed supply points in S only and totally ignore the demand points in D. In Figure 2.2 we show an example. In this example, there exists a

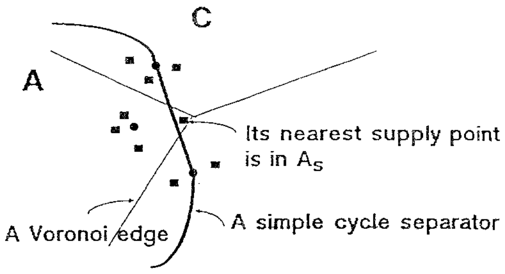

Fig. 2.2. An example of a Delaunay triangulation graph. ■, demand points; ●, supply points.

demand point lying in C_d and its closest supply point is in A_s, if we treat the simple cycle separator as the desired B-cycle.

In the following we define the rules which construct the B-cycle from the simple cycle separator B. Let Vor(S') denote the Voronoi diagram of S' (Voronoi, 1907; Preparata and Shamos, 1985; Edelsbrunner, 1987) and let DT(S') denote the Delaunay triangulation of S' (Delaunay, 1934; Preparata and Shamos, 1985; Edelsbrunner, 1987). Given a simple cycle separator B on DT(S'), the corresponding B-cycle is defined as follows: for every connected pair of supply points s_i and s_j in the simple cycle separator B, find its associated edge e in the Vor(S'). (Note that for any edge on the DT(S'), there is an associated edge on Vor(S').) Let one of the two points of the edge e be v_{ij}. Draw $\overline{s_i v_{ij}}$ and $\overline{v_{ij} s_j}$. This new corresponding cycle, consisting of all such $\overline{s_i v_{ij}}$'s and $\overline{v_{ij} s_j}$'s, is called the B-cycle of S'.

Consider Figure 2.3. Figure 2.3(a) shows a simple cycle separator and Figure 2.3(b) shows the corresponding B-cycle.

It is obvious that the B-cycle, constructed by the above rules, satisfies the first property, that is, the number of unfixed supply points in the B-cycle is less than $\sqrt{8 \cdot (P + 3)}$. Now we want to show the second property. From Miller's simple cycle separator, we know that the sets of unfixed supply points divided by the simple cycle separator satisfy the second property. Now we want to show, in the following lemma, that the sets of unfixed supply points divided by a simple cycle separator B are identical to that divided by its corresponding B-cycle. This way we can prove the second property.

Lemma 2.1. *The sets of unfixed supply points in the interior and exterior parts partitioned by the simple separator cycle B and its corresponding B-cycle are identical.*

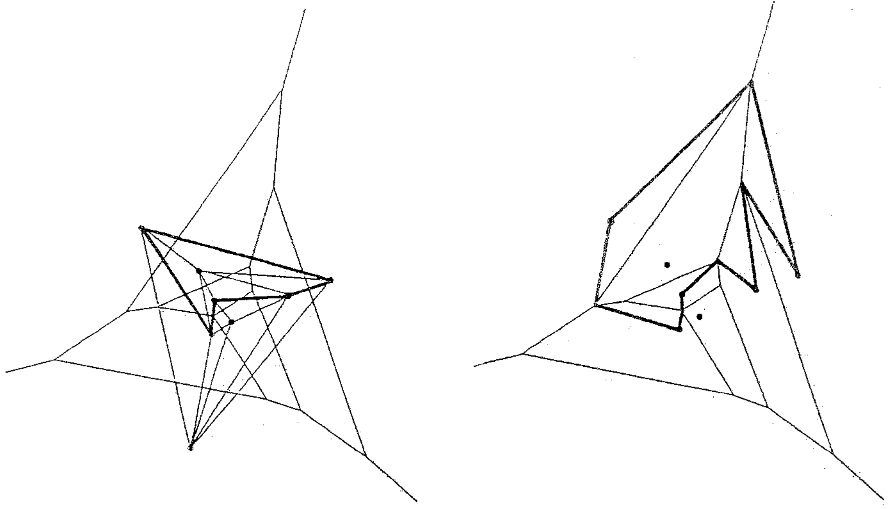
(a) (b)

Fig. 2.3. (a) A simple cycle separator and (b) the corresponding B-cycle.

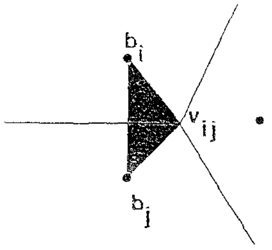

Fig. 2.4. No unfixed supply point in the shaded area.

Proof. Consider any edge $\overline{s_i s_j}$ in the separator cycle B. Let e be the Voronoi edge associated with $\overline{s_i s_j}$. Let v_{ij} be one of the vertices on e. Consider s_i, s_j, and v_{ij}, as shown in Figure 2.4. According to Theorem 5.8 of Preparata and Shamos (1985), there is no unfixed supply point in this triangle. Therefore, if we replace $\overline{s_i s_j}$ by $\overline{s_i v_{ij}}$ and $\overline{v_{ij} s_j}$, the partition of unfixed supply points is not changed. □

The last and most important property is the third one. Now we want to show that, for each demand point in A_d (resp. C_d), its nearest supply point is in A_s (resp. C_s) or $B_s \cup \beta$.

Theorem 2.2. *Given an instance of* GDEPM(P, D, β) *or* GDEPC(P, D, β), *its optimal solution* S, *and the corresponding B-cycle, let* B_s *be the set of supply points on the B-cycle. The B-cycle divides* D *(resp.* S*) into interior and exterior two parts, called* A_d *and* C_d *(resp.* A_s *and* C_s*). Then, for each demand point in* A_d *(resp.* C_d*), its nearest supply point is in* A_s *(resp.* C_s*) or* $B_s \cup \beta$.

Proof. To show that, for each demand point in A_d (resp. C_d), its nearest supply point belongs to A_s (resp. C_s) or $\beta \cup B_s$, we first use the fact that no edge in the B-cycle crosses $V(s')$ where $s' \notin B_s$ (where $V(s)$ denotes the Voronoi polygon associated with s, $s \in S$). This fact is due to the rules constructing the B-cycle.

Let $d \in A_d$ and let the nearest supply point of d be $s \in C_s$. Hence d must be on $V(s)$, if Vor($S \cup I$) is constructed, where I is the set of enclosing points. Because d is in the interior part of the B-cycle and s is in the exterior part, there must exist some edges of the B-cycle passing through $V(s)$. However, from the above fact of the B-cycle, the edges can only cross the polygon associated with the unfixed supply points in B_s. Therefore no such a demand point d exists. This means that the nearest supply point of any demand point in A_d (resp. C_d) must be in A_s (resp. C_s) or $B_s \cup \beta$. □

The above theorem certifies that the two subproblems are independent. This independent property is very important because it guarantees the correctness of our searching over separators approach.

Now let us show a simple example to explain how to divide the input points by using the B-cycle. In this example we assume that S' is given as in Figure 2.5 and P is 18. The largest size of the B-cycle is therefore at most $\sqrt{8 \cdot (18 + 3)}$. In

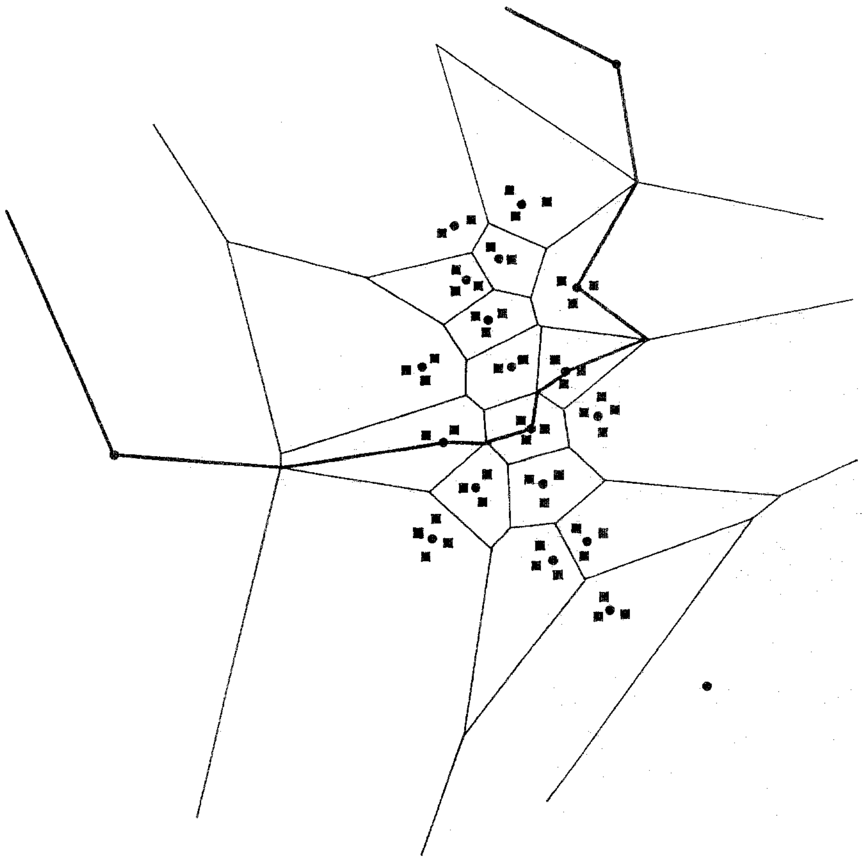

Fig. 2.5. An example of how to divide input points by using a B-cycle. ■, demand points; ●, supply points.

Figure 2.5 there are a set of demand points (denoted as squares), a set of unfixed supply points (drawn as dots), the enclosing points, and the B-cycle of the Delaunay triangulation of S', constructed out of two enclosing points and four unfixed supply points. The B-cycle divides the points into two parts. We can see that, for each demand point in the interior (resp. exterior) part of the B-cycle, its nearest supply point is either in the interior (resp. exterior) or on the B-cycle.

2.3. Generating Possible B-Cycles. In the preceding subsection we showed the properties and the existence of a B-cycle. In this section we introduce two procedures, which will generate possible B-cycles, one of which is the desired B-cycle.

Let us first discuss the problem of generating a simple cycle separator. A simple cycle separator consists of less than $\sqrt{8(P+3)}$ points. We exhaustively try all possible ways of selecting i points out of $n+3$ points (including the three enclosing points), where i ranges from 3 to $\sqrt{8(P+3)}$ (for a simple cycle needs at least three points). (Thus, there are at most

$$O\left(\binom{(n+3)}{\sqrt{8(P+3)}}\cdot\sqrt{8(P+3)}\right)$$

ways to select i points.) For each set of i points, we construct a complete graph. Out of this complete graph, we select any i edges and test whether they form a simple cycle or not. (Since there are $i\cdot(i-1)$ edges in the graph, there are at most $O\left(\binom{i\cdot(i-1)}{i}\right)$ possible selections. To test whether they form a simple cycle needs at most $O(i^2)$.) One of these cycles must be the desired simple cycle separator. From the above discussion, we know that the time needed to generate all possible simple cycle separators is

$$O\left(\binom{(n+3)}{\sqrt{8(P+3)}}\cdot\sqrt{8(P+3)}\binom{i\cdot(i-1)}{i}\cdot i^2\right)=O(n^{c_1\sqrt{P}+c_2}),$$

where c_1 and c_2 are some constants.

Next, we should try to construct the corresponding B-cycle of a given simple cycle separator by using the rules defined in Section 2.2. Those rules tell us that we can construct the B-cycle by connecting all such $\overline{s_i v_{ij}}$'s and $\overline{v_{ij} s_j}$'s for each edge $\overline{s_i s_j}$ in the simple cycle, where v_{ij} is the Voronoi polygon shared by $V(s_i)$ and $V(s_j)$, where $V(s_i)$ (resp. $V(s_j)$) is the Voronoi polygon associated with s_i (resp. s_j). We cannot find v_{ij} directly, for $V(s_i)$ and $V(s_j)$ are unknown. Therefore we propose an exhaustive search approach to find all possible candidates of v_{ij}. The Voronoi vertex v_{ij} must be the center of the circle formed by s_i, s_j and the third unknown point s_k. We know that $s_k \in S \cup I$. Therefore $s_k \in D \cup I$. We may try all points in $D \cup I$ as the candidates of s_k. This way, for each edge $\overline{s_i s_j}$ in the simple cycle, we have $|D \cup I| = (n+3)$ possible v_{ij}'s, by finding the center of the circumscribed circles defined by the three points s_i, s_j, and any point in $D \cup I$.

Consider the time complexity of the above steps. Step 1 takes

$$\binom{n+3}{i+1}\cdot(i+1)! \le (n+3)^{i+1}$$

steps. Steps 2–8 take $O(i^2)$ time. The time needed in this procedure is

$$O((n+3)^{i+1}\cdot i^2).$$

Since $i+1$ is the number of points in the simple cycle and $(i+1) \le \sqrt{8(P+3)}$, we can see that the time complexity is bounded by $O(n^{c_3\sqrt{P}+c_4})$, where c_3 and c_4 are some constants.

In the next section we show the entire algorithm to solve the GDEPM problem, by using the above two procedures.

2.4. The Algorithm and Its Time Complexity. In this section we present an algorithm to solve the GDEPM problem. This algorithm is called Algorithm GDEPM.

Algorithm GDEPM(P, D, β, S, C). (An algorithm to solve the GDEPM problem based upon the search over separators strategy.)
Input: D, a set of n demand points; β, a set of fixed supply points; and P; a number.
Output: S, a set of supply points which is an optimal solution to the GDEPM problem; C, the optimal cost of the GDEPM problem.

```
Step 1.  Let $C = \infty$.
Step 2.  If $P \le 3$, then do steps 3–5:
Step 3.     For each subset $S' \subset D$ of $P$ points do:
Step 4.        For all points in $D$, find their corresponding nearest points
               in $S' \cup \beta$ and sum up all these weighted distances to $C'$.
Step 5.        If $C > C'$, then $C = C'$ and $S = S'$.
Step 6.  Else do steps 7–13:
Step 7.     Generate all possible B-cycles (using the method discussed in
            the above section).
Step 8.        For each possible B-cycle, we divide $D$ into the interior
               part $A'_d$ and the exterior part $C'_d$, and do steps 9-13:
Step 9.           For $k = 0$ to $2P/3$ do:
Step 10.             If $(P - i - k) \le 2P/3$ do:
Step 11.                Call GDEPM($k$, $A'_d$, $B'_s \cup \beta$, $S_1$, $C_1$).
Step 12.                Call GDEPM($P - i - k$, $C'_d$, $B'_s \cup \beta$, $S_2$, $C_2$).
Step 13.                If $C_1 + C_2 < C$, then $S = S_1 \cup S_2 \cup B'_s$ and
                        $C = C_1 + C_2$.
```

Now let us discuss the time complexity of Algorithm GDEPM. Let the total time complexity of this algorithm be $T(P)$. We can see that steps 2–5 take $O(n^{P+1} \cdot P)$, $P \le 3$. Step 7 needs $O(n^{(c_1\sqrt{P}+c_2)+(c_3\sqrt{P}+c_4)}) = O(n^{c_5\sqrt{P}+c_6})$, as described in the above section, where $c_5 = c_1 + c_3$ and $c_6 = c_2 + c_4$. Steps 9–13 are bounded by $O((2P/3) \cdot 2 \cdot T(2P/3)) \le O(2n \cdot T(2P/3))$, for $P \le n$. Therefore the time complexity from step 7 to step 13 is bounded by $O(n^{c_5\sqrt{P}+c_6+1}) \cdot T(2P/3)$. Hence we have the following formula:

$$T(P)\begin{cases} \le O(n^{c_5\sqrt{P}+c_6+1}) \cdot T(2P/3) & \text{when } P > 3, \\ = O(n^{P+1} \cdot P) \le O(n^4) & \text{when } P \le 3. \end{cases}$$

When $P > 3$,

$$\begin{aligned} T(P) &\le O(n^{c_5\sqrt{P}+c_6+1}) \cdot T(2P/3) \\ &\le O(n^{c_5\sqrt{P}+c_6+1}) \cdot O(n^{c_5\sqrt{2P/3}+c_6+1}) \cdot T(4P/9)) \\ &\le O(n^{c_5(\sqrt{P}+\sqrt{2P/3}+\sqrt{4P/9}+\cdots)+(c_6+1)\cdot\log_{3/2}P}) \\ &\le O(n^{c_5\cdot(1/(1-\sqrt{2/3}))\cdot\sqrt{P}+(c_6+1)\cdot\log_{3/2}P}) = O(n^{O(\sqrt{P})}). \end{aligned}$$

Therefore we conclude that $T(P) = O(n^{O(\sqrt{P})})$.

2.5. The Discrete Euclidean P-Center Problem. After showing how the DEPM problem can be solved by the searching over separators strategy, in this section we show that the DEPC problem can be solved in exactly the same way.

The DEPC problem is defined as follows: given a set $D = \{d_1, d_2, \ldots, d_n\}$ of demand points, find a set S of P supply points from D, in order to minimize

$$\max_{d_i \in D} \left\{ \min_{s_j \in S} \{l(d_i, s_j)\} \right\}.$$

Megiddo and Supowit (1984) proved that the EPC problem is NP-hard. Drezner proposed an $O(n^{2P+1} \cdot \log n)$ algorithm (Drezner, 1984) for the EPC problem, which can be improved to $O(n^{2P-1} \cdot \log n)$ by combining it with the result that the Euclidean 1-center problem can be solved in $O(n)$ time (Megiddo, 1983). The way of combining these two results is similar to that of Drezner (1987).

We can see that the DEPC problem is almost exactly the same as the DEPM problem. We need to define *a generalized discrete Euclidean P-center problem* as follows: given a set D of n demand points, a set $\beta \subset D$ of fixed supply points, and a number P, select a set S of P supply points from D where β and S are disjoint, such that

$$\max_{d_i \in D} \left\{ \min_{s_j \in S \cup \beta} \{l(d_i, s_j)\} \right\}$$

is minimized.

We may denote the above problem as the GDEPC-(P, D, β) problem or the GDEPC problem. Thus, if we change step 13 of algorithm DEPM to be as follows:

Step 13. If $\max\{C_1, C_2\} < C$, then $S = S_1 \cup S_2 \cup B'_s$ and $C = \max\{C_1, C_2\}$.

then algorithm DEPM can also be used to solve the DEPC problem with time complexity $O(n^{O(\sqrt{P})})$.

3. The Euclidean *P*-Center Problem

3.1. Drezner's Algorithm. The EPC problem is similar to the DEPC problem, except in this case the supply points do not have to be chosen from demand points. Yet, by using Drezner's algorithm (Drezner, 1984), we can easily transfer this problem to a problem similar to the DEPC problem and thus the search over separators strategy can be used again.

Note that P circles of radius r can cover n demand points if and only if there are P supply points and the longest distance between each demand point and its closest supply points is r. Drezner (1984) pointed out that a circle is defined by one, two, or three points. For the case of circles defined by three points, these three points define the boundary of the smallest circle enclosing all three of them.

For the case defined by two points, they form the diameter of this circle. A circle defined by only one point is a degenerated case, where the radius of this circle can be considered as zero and the entire circle contracts to one point. Hence, we can simply enumerate all of the circles. There are $\binom{n}{1}$, $\binom{n}{2}$, and $\binom{n}{3}$ circles defined by one, two, and three points, respectively. We call these circles *the bounding circles.*

Given any solution to the P-center problem, the largest radius r (the longest of all distances between each demand point and its closest supply point) must be equal to the radius of one of the bounding circles bounded by two or three points. Hence, if we have an algorithm which can determine whether P circles of a given radius r can cover all these points, we can perform a binary search over all radii of the bounding circles to find an optimal one.

Let us sort all of the radii of the bounding circles. Then we choose one of them, say r, and we ask the following question: can P circles of radius r cover the n points? To answer this question, apparently we have to determine where these P circles should be placed.

Drezner (1984) also showed that there are only $O(n^2)$ possible supply points for a given radius r. He claimed that there exists a set S_p of $O(n^2)$ possible supply points, such that if P circles of radius r can cover the n demand points, then we can find P circles centered at some of the points in S_p which can cover the n demand points. The points in set S_p are called the possible supply points, found as follows: For any two points in the n demand points, we find the two circles of radius r passing through these two points. The set S_p is the union of the centers of these circles and the n demand points.

Now let us see why Drezner's claim holds. Assume that we have found that a set $C = \{c_1, c_2, \ldots, c_P\}$ of circles of radius r can cover all n demand points. If there are more than two demand points on the boundary of circle $c_i \in C$, then the center of c_i belongs to S_p. If there are less than two demand points on the boundary of circle $c_j \in C$, then there are two possibilities. One is that this circle covers only one demand point. In this case we move this circle so that it centers at this demand point. Another case is that this circle covers more than two demand points. In this case we can move this circle until two demand points touch the boundary of this circle. Again this new center belongs to S_p. We can see that these new circles cover the same set of demand points as the old circles do. So we have a new solution and the centers of the circles in this solution are all points in S_p.

Therefore we can select any P points in S_p and then check whether these circles of radius r centered at these P points cover all of the n demand points. Since there are $O(n^2)$ possible supply points, we have $\binom{O(n^2)}{P}$ possible selections, and it takes $O(n)$ time to check whether these P circles cover all points, and $O(\log n^3) = O(\log n)$ to do the binary search. So the time complexity is $O(n^{2P+1} \cdot \log n)$.

3.2. The Searching over Separators Strategy To Solve the (P, D, S_p) Circle Cover Optimization Problem. As we discussed above, our basic problem is as follows: given a set D of n demand points, a radius r, and a set S_p of $O(n^2)$ possible supply points (generated by using Drezner's algorithm (Drezner, 1984)), determine whether there exists a set $S \subset S_p$ of P supply points, such that the P circles of radius r centered at these P points can cover the n demand points. We call this problem *the (P, D, r, S_p) circle cover decision problem* (the (P, D, r, S_p) CCD problem).

In order to solve the above (P, D, r, S_p) CCD problem, we may first solve the following (P, D, S_p) *circle cover optimization problem* (the (P, D, S_p) CCO problem): given a set D of n demand points and a set S_p of $O(n^2)$ possible supply points, find the smallest r_s, such that P circles of radius r_s centered at some P points selected from S_p can cover the n demand points. We can see that if this r_s is longer than r, then the answer to the (P, D, r, S_p) CCD problem is "false"; otherwise it is "true." Thus, if the (P, D, S_p) CCO problem is solved, the (P, D, r, S_p) CCD problem is also solved.

It can be easily seen that the (P, D, S_p) CCO problem is similar to the GDEPC-(P, D, β) problem except in this case there are $O(n^2)$ points in S_p and we select the supply points from S_p, instead of D. Thus the algorithm for solving the GDEPC problem can be used to solve the (P, D, S_p) CCO problem with time complexity $O(n^{O(\sqrt{P})})$.

4. The Euclidean Traveling Salesperson Problem

4.1. Preliminaries. In the above sections we described how to apply the searching over separators strategy to the DEPC problem. We now show how the searching over separators strategy can be applied to solve *the Euclidean traveling salesperson (ETSP) problem.* In this problem, given n points in the plane, we are asked to find a shortest cycle out of these n points. This problem is an NP-hard problem (Papadimitriou, 1977; Papadimitriou and Steiglitz, 1976) and it can be solved by the dynamic programing strategy in time $O(2^n n^2)$ (Held and Karp, 1962; Horowitz and Sahni, 1978; Lawler *et al.*, 1985). A very thorough review of many algorithms for this problem can be found in Lawler *et al.* (1985). We show that through the searching over separators strategy, we can obtain an algorithm for the ETSP problem with $O(n^{O(\sqrt{n})})$ time.

4.2. The Generalized Euclidean Traveling Salesperson Problem. As we did in solving the DEPM problem, we first generalize our ETSP problem into the following: Given a set $V = \{v_1, v_2, \ldots, v_n\}$ of points and a set

$$T = \{(t_1, t'_1), (t_2, t'_2), \ldots, (t_m, t'_m)\}$$

of *terminal pairs*, we want to find a set of m paths, satisfying the following conditions:

(1) Every path starts from t_i and returns to t'_i, where $1 \le i \le m$.
(2) Every point in V is included in exactly one of these paths.
(3) The total length of these paths is minimized.

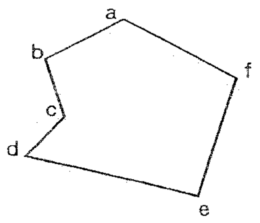

Fig. 4.1. An optimal solution to the ETSP problem.

We may call *the generalized Euclidean traveling salesperson problem* (the GETSP problem) with such V and T as inputs *the* (V, T)*-GETSP problem.* When $V = \{v_2, v_3, \ldots, v_n\}$ and $T = \{(v_1, v_1)\}$, this GETSP problem is degenerated to the ETSP problem with $v_1, \ldots, v_n$ as inputs. Again, for reasons which will become clear later, we try to solve the GETSP problem.

Let us imagine that given an ETSP problem with a set V of n points as input and we have an optimal solution to this problem. We first show that, based upon the optimal solution, we can find a cycle separator properly dividing the input sets V into two parts of inputs V_a, T_a and V_c, T_c of the GETSP problems such that, if we solve the two GETSP problems and then merge the solutions, we can obtain an optimal solution of the original one.

Consider Figure 4.1, which shows a set of points and an optimal solution to the ETSP problem. We then add a set $I = \{I_1, I_2, I_3\}$ of *three enclosing points* which inscribe all points in V, and construct a triangulation out of the points in $V \cup I$ where every edge in the optimal solution is also an edge in this triangulation. Since this triangulation is a maximal planar graph, let the points in I be zero weighted vertices and let the others be equal weighted vertices. Then from Miller's simple cycle separator theorem (Miller, 1986), there exists a simple cycle separator which divides the points in V into the interior part V_a and the exterior part V_c, respectively, such that there are no more than $\sqrt{8 \cdot (n + 3)}$ vertices in the simple cycle separator, $|V_a| \leq 2n/3$ and $|V_c| \leq 2n/3$, where n is the number of points in V and in this case $n = 6$. Here the three input points, namely, a, c, and e are on the simple cycle separator, as shown in Figure 4.2. Thus we now have two GETSP problems, defined as follows:

(1) The interior subproblem with $V_a = \{f\}$, $T_a = \{(e, a)\}$ as inputs.
(2) The exterior subproblem with $V_c = \{b, d\}$, $T_c = \{(a, c), (c, e)\}$ as inputs.

After solving the above two GETSP problems, we obtain the three paths, $p_1 = (a, b, c)$, $p_2 = (c, d, e)$, and $p_3 = (e, f, a)$. Using a simple merging process, we can derive the original optimal solution $p = (a, b, c, d, e, f, a)$.

The above example shows how to divide the ETSP problem into two GETSP problems. For solving the subproblems, we would recursively divide these GETSP problems. It is not difficult to see that the GETSP problem can also be divided by the same principle, except that there is an extra input data T. In this new dividing process, we construct the triangulation out of the points in V, I, and T. Also let the points in I be zero weighted vertices and let the others be equal

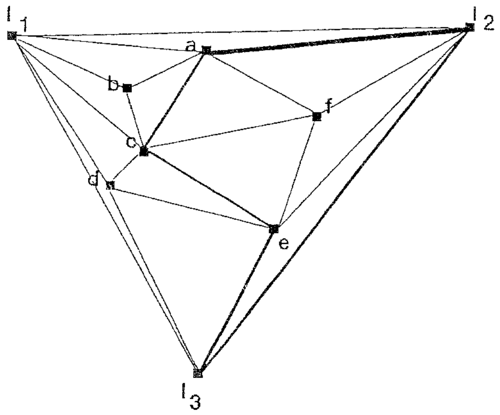

Fig. 4.2. A simple cycle separator showing the three input points a, c, and e.

weighted vertices. Then, from Miller's simple cycle separator theorem (Miller, 1986), there exists a simple cycle which divides the points in V and T into the interior parts V_a, T'_a and the exterior parts V_c, T'_c, respectively, such that there are no more than $\sqrt{8 \cdot (n + 2m + 3)}$ vertices in the simple cycle and $|V_a| + |T'_a| \leq 2(n + 2m)/3$, $|V_c| + |T'_c| \leq 2(n + 2m)/3$, where $2m$ is the number of points in T.

In the following we present a dividing process, which would decompose a (V, T)-GETSP problem into two subproblems. We later show that if we solve these two subproblems and then merge the two solutions, we would obtain an optimal solution to the original (V, T)-GETSP problem. As explained later, to execute this dividing process we must have an optimal solution to the (V, T)-GETSP problem.

A Process To Divide a (V, T)-GETSP Problem into Two Subproblems

Input: A (V, T)-GETSP problem, where $V = \{v_1, v_2, \ldots, v_n\}$ and $T = \{(t_1, t'_1), (t_2, t'_2), \ldots, (t_m, t'_m)\}$, and an optimal solution to this problem, namely, a set S of m paths $p_1, p_2, \ldots, p_m$.

Output: Two subproblems: the (V_a, T_b)-GETSP problem and the (V_c, T_c)-GETSP problem, such that the merging of the solutions to the above two problems will result in an optimal solution to the (V, T)-GETSP problem.

Step 1. Find the set $I = \{I_1, I_2, I_3\}$ of three points which inscribes all points in V and T.

Step 2. Construct a triangulation graph out of these points in V, T, and I, such that each edge in the optimal solution is an edge in this triangulation.

Step 3. Let the weights of points in I be zero and let others be any nonzero constant. Use a simple cycle separator B to divide V (resp. the points in T) into the interior part V_a (resp. T'_a) and the exterior part V_c (resp. T'_c). Let V_b be the points on the simple cycle separator B which belong to V.

```
Step 4. Let $T_a = T_c = \varnothing$.
Step 5. For $i = 1$ to $m$ do:
Step 6.     Call Procedure
            DIVIDE_TERMINAL($p_i$, $V_a$, $V_c$, $T'_a$, $T'_c$, $V_b$, $T^i_a$, $T^i_c$).
Step 7.     $T_a = T_a \cup T^i_a$, $T_c = T_c \cup T^i_c$.
Step 8. Output two subproblems: the $(V_a, T_a)$-GETSP problem and the
        $(V_c, T_c)$-GETSP problem.
```

In the above process the critical procedure is Procedure DIVIDE_TERMINAL. This is a linear scan procedure, which scans the points along a path. If the point being scanned is a point in the simple cycle separator, then some appropriate action is taken. This procedure is now described:

```
Procedure DIVIDE_TERMINAL($p_i$, $V_a$, $V_c$, $T'_a$, $T'_c$, $V_b$, $T^i_a$, $T^i_c$)
Input: A path $p_i = (v_{i1}, v_{i2}, \ldots, v_{ij})$, $V_a$, $V_c$, $T'_a$, $T'_c$, and $V_b$.
Output: $T^i_a$ and $T^i_c$ where $T^i_a$ and $T^i_c$ are both sets of terminal pairs.

Step 1. $v' = v_{i1}$, $T^i_a = T^i_c = \varnothing$.
Step 2. For $k = 2$ to $j$ do:
Step 3.     If $v_{ik} \in V_b$ or $k = j$, then:
Step 4.         If $(v' \in T'_a)$ or $(v_{ik} \in T'_a)$ or $(v_{i(k-1)} \in V_a)$ then:
Step 5.             $T^i_a = T^i_a \cup \{(v', v_{ik})\}$
Step 6.         Else
                    $T^i_c = T^i_c \cup \{(v', v_{ik})\}$
Step 7.     $v' = v_{ik}$.
```

In the following we show an example to illustrate how Procedure DIVIDE_-TERMINAL works. Consider Figure 4.3. We have a path $p_i = (a, b, c, d, e, f, g, h)$, where $a, h \in T'_c$, $b, e, g \in V_b$, $c, d \in V_a$, and $f \in V_c$. Applying the above procedure, we obtain $T^i_c = \{(a, b), (e, g), (g, h)\}$ and $T^i_a = \{(b, e)\}$. We now show that after we solve the two subproblems, we can merge the two solutions and obtain an optimal solution to the original problem.

Note that in the dividing process, Procedure DIVIDE_TERMINAL sequentially divides a path into sets of terminal points. Thus, for path p_i, let us assume that the terminal pairs resulting in both interior and exterior parts from applying the above procedure are $(t_{i1}, t'_{i1}), (t_{i2}, t'_{i2}), \ldots, (t_{ij}, t'_{ij})$, where t_{i1} and t'_{ij} are the starting and terminating points of p_i, respectively. After solving the (V_a, T_a)-GETSP

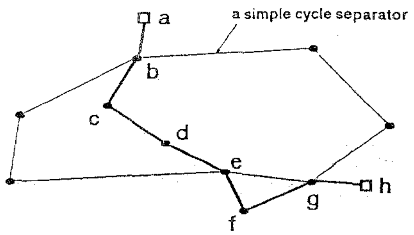

Fig. 4.3. Path $p_i = (a, b, c, d, e, f, g, h)$, where $a, h \in T'_c$, $b, e, g \in V_b$, $c, d \in V_a$, and $f \in V_c$.

problem and the (V_c, T_c)-GETSP problem, t_{i1} is connected to t'_{i1} through a path, t_{i2} is connected to t'_{i2} through a path, and so on. Moreover, t_{i1}, which is a starting point of p_i, is connected to t'_{uj}, which is a terminating point of p_i. In other words, after solving the (V_a, T_a)-GETSP problem and the (V_c, T_c)-GETSP problem, we have $p'_1, p'_2, p'_3, \ldots, p'_m$ in which the starting point and terminating point of p'_i are the same as those of p_i of the optimal solution to the (V, T)-GETSP problem. We may conclude that after solving the (V_a, T_a)-GETSP problem and the (V_c, T_c)-GETSP problem, we have obtained a solution to the (V, T)-GETSP problem. We now prove that this solution is an optimal solution.

Let S_a and S_c denote optimal solutions to the (V_a, T_a)-GETSP problem and the (V_c, T_c)-GETSP problem, respectively. Let S denote an optimal solution to the (V, T)-GETSP problem, and let COST(S) denote the cost of solution S. We show that $\text{COST}(S_a) + \text{COST}(S_c) = \text{COST}(S)$. Assume otherwise. Then there are two cases:

Case 1: $\text{COST}(S_a) + \text{COST}(S_c) < \text{COST}(S)$. This is impossible because COST(S) is the cost of an optimal solution to the (V, T)-GETSP problem.

Case 2: $\text{COST}(S_a) + \text{COST}(S_c) > \text{COST}(S)$. Let us decompose the optimal solution into two parts, one relating to points in the (V_a, T_a)-GETSP problem, denoted as S'_a, and the other relating to points in the (V_c, T_c)-GETSP problem, denoted as S'_c. Then $\text{COST}(S) = \text{COST}(S'_a) + \text{COST}(S'_c)$. In other words, we have

$$\text{COST}(S_a) + \text{COST}(S_c) > \text{COST}(S'_a) + \text{COST}(S'_c).$$

Without loss of generality, we may assume that $\text{COST}(S_a) > \text{COST}(S'_a)$. Again, this is impossible because COST(S_a) corresponds to the cost of an optimal solution to the (V_a, T_a)-GETSP problem.

In conclusion, we have the following lemma:

LEMMA 4.1. *Let S_a and S_c denote the costs of optimal solutions to the (V_a, T_a)-GETSP problem and (V_c, T_c)-GETSP problem, respectively, and let S denote an optimal solution to the (V, T)-GETSP problem. Then*

$$\text{COST}(S) = \text{COST}(S_a) + \text{COST}(S_c).$$

There is still one problem which we have to solve. Note that after solving two (V, T)-GETSP problems, we get a set of paths. These paths cannot intersect with one another because Miller's theorem can only be applied to the planar graph. It is obvious that there is no intersection in the optimal solution to the ETSP problem. Assume that S is an optimal solution to some ETSP problem, $\overline{v_1v_2}$ and $\overline{v_3v_4}$ are in S, and they intersect each other, then we can draw $\overline{v_1v_3}$, $\overline{v_2v_4}$ or $\overline{v_1v_4}$, $\overline{v_2v_3}$ instead of $\overline{v_1v_2}$, $\overline{v_3v_4}$ in S. Both will derive a lower cost than the original one and one of them must be a legal solution. So we have another legal solution with lower cost than S. Therefore any cycle with intersections must not be an optimal solution to the ETSP problem.

We next show that if the initial problem is an ETSP problem, then at each stage after solving two GETSP problems, the resulting paths do not intersect and

Miller's theorem can always be applied. Assume otherwise. At a certain stage, two paths intersect after solving two GETSP problems. Then after we merge these paths back recursively, we finally obtain a cycle with intersections. This is impossible because our initial problem is an ETSP problem, and its solution should be a cycle without intersections.

4.3. Generating All Possible Input Instances. Note that the above discussion is based upon an assumption that an optimal solution is available to us. Of course, we do not know any optimal solution in advance. Therefore our strategy is to generate all possible simple cycle separators, and then we find the best result with respect to each possible simple cycle separator. Because the largest size of a simple cycle separator is $\sqrt{8(n+2m+3)}$, we may select i edges, where $3 \le i \le \sqrt{8\cdot(n+2m+3)}$, from the complete graph constructed from the points in V, T, and I. If these i edges form a simple cycle without any intersection, then test whether this cycle can divide the rest of the points in V, T into V_a, T'_a and V_c, T'_c, such that $|V_a \cup T'_a| \le 2(n+2m)/3$ and $|V_c \cup T'_c| \le 2(n+2m)/3$. If this cycle satisfies the above conditions, this is a candidate of the simple cycle separator. Since we exhaustively generate all possible such simple cycles, the one which derives the best result must be the desired one. The following is our algorithm to generate all these possible simple cycle separators.

```
Procedure GEN_CYCLES_B($V$, $T$, $B'$)
Function: Generate a set $B'$ of simple cycles of which one is the simple
cycle separator on the maximal planar graph from the points in $V$, $T$,
and $I$, and the edges in the paths of an optimal solution.
Input: A set $V$ and $T$.
Output: A set $B'$ of simple cycles.

Step 1.  Let $B' = \emptyset$ and let $T'$ be the set of points in $T$.
Step 2.  For $i = 3$ to $\sqrt{8(|V| + |T'| + 3)}$ do:
Step 3.  Find $B'_s = \{b'_s | b'_s \subset (V \cup T' \cup I)$ and $|b'_s| = i\}$.
Step 4.  For each $b'_s \in B'_s$ do:
Step 5.     Construct the complete graph out of the points in $b'_s$.
Step 6.     For each subset of $i$ edges in $G$ do:
Step 7.        If these $i$ edges cannot form a simple cycle, then jump to
               step 6 and try the next instance; else denote this simple
               cycle as $b$.
Step 8.        If there exists any intersection in $b$, then jump to step 6 and
               try the next instance.
Step 9.        Use $b$ to divide the points in $V$ and $T$ into interior parts
               $V_a$, $T'_a$ and the exterior parts $V_c$, $T'_c$. (Apply the point
               location algorithm of Preparata and Shamos (1985).) If
               $(|V_a \cup T'_a| > 2(n + 2m)/3)$ or $(|V_c \cup T'_c| > 2(n + 2m)/3)$, then
               jump to step 6 and try the next instance.
Step 10.       $B' := B' \cup b$.
Step 11. Return $B'$.
```

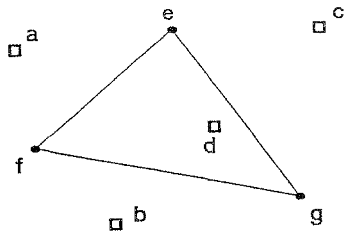

Fig. 4.4. Two terminal pairs, (a, b) and (c, d), and three points, e, f, and g, on the simple cycle separator.

This procedure is slightly modified from Procedure GEN_CYCLES_A discussed in the preceding sections. By using a similar method, we can prove that the time complexity and the number of cycles are both bounded by $O((n + 2m)^{c_a \cdot (\sqrt{n+2m}) + c_b})$, where c_a and c_b are some constants.

In this procedure we show that we can find all possible simple cycle separators. We still have to face one problem. We have to decide all of the terminal pairs. For instance, in the example illustrating the dividing procedure, we generate two sets of terminal pairs $T_c^i = \{(a, b), (e, g), (g, h)\}$ and $T_a^i = \{(b, e)\}$. Since no optimal solution is available to us, how can we determine these two sets?

Let us consider Figure 4.4. In Figure 4.4 there are two terminal pairs (a, b) and (c, d). The points on the simple cycle separator are e, f, and g. Of course, there are many other points which are neither terminal points nor points on the simple cycle separator. From Procedure DIVIDE_TERMINAL, it can be seen that points which are neither old terminal points nor points on the simple cycle separator are irrelevant to our dividing process. Therefore, in the following, we only consider points which are originally terminal points or points on the simple cycle separator.

Suppose that we are given two terminal pairs (a, b) and (c, d). We are also given points on the simple cycle separator e, f, and g. There are many ways of forming paths by using these points. Our job means to assign e, f, and g to either (a, b) or (c, d) such that two paths will be formed: one connecting a to b and the other connecting c to d. In the following we list some of them:

(1)	(a, e, b)	and	(c, f, g, d)
(2)	(a, e, b)	and	(c, g, f, d)
(3)	(a, b)	and	(c, e, f, g, d)
(4)	(a, b)	and	(c, f, e, g, d)
(5)	(a, f, g, b)	and	(c, e, d)
(6)	(a, g, f, b)	and	(c, e, d)

and so on.

It can be seen that in each case we assign a subset of e, f, and g to (a, b) and the rest of the points to (c, d). Note that the subset can be an empty set. Moreover, after a subset of points from e, f, and g are assigned, they can be permuted.

For each set of paths, there is a corresponding set of terminal pairs. Again, in the following we list some of them:

(1)	(a, e, b)	$\rightarrow$	$(a, e), (e, b)$
	(c, f, g, d)	$\rightarrow$	$(c, f), (f, g), (g, d)$
(2)	(a, e, b)	$\rightarrow$	$(a, e), (e, b)$
	(c, g, f, d)	$\rightarrow$	$(c, g), (g, f), (f, d)$
(3)	(a, b)	$\rightarrow$	(a, b)
	(c, e, f, g, d)	$\rightarrow$	$(c, e), (e, f), (f, g), (g, d)$
(4)	(a, b)	$\rightarrow$	(a, b)
	(c, f, e, g, d)	$\rightarrow$	$(c, f), (f, e), (e, g), (g, d)$
(5)	(a, f, g, b)	$\rightarrow$	$(a, f), (f, g), (g, b)$
	(c, e, d)	$\rightarrow$	$(c, e), (e, d)$
(6)	(a, g, f, b)	$\rightarrow$	$(a, g), (g, f), (f, b)$
	(c, e, d)	$\rightarrow$	$(c, e), (e, d)$

For each terminal pair, we assign it to interior or exterior. For instance, consider (a, e). Since a is an exterior point, (a, e) is an exterior terminal pair. On the other hand, (g, d) is an interior terminal pair, because d is an interior point. For (f, g), it can be assigned to either exterior or interior. Therefore, given the set of terminal pairs $\{(a, e), (e, b), (c, f), (f, g), (g, d)\}$, we can generate the following possible terminal pairs of the subproblems:

(1) The interior terminal pairs: $\{(c, f), (f, g)\}$.
  The exterior terminal pairs: $\{(a, e), (e, b), (g, d)\}$.
(2) The interior terminal pairs: $\{c, f)\}$.
  The exterior terminal pairs: $\{(a, e), (e, b), (f, g), (g, d)\}$.

We now list the basic steps to generate all the new terminal pairs. We are given the original terminals $(t_1, t'_1), (t_2, t'_2), \ldots, (t_m, t'_m)$ and the set of nonterminal points in the simple cycle separator $\{b_1, b_2, \ldots, b_k\}$. Our first step is to generate all possible permutations of $\{b_1, b_2, \ldots, b_k\}$. For instance, for the above example, we have

$$\begin{array}{c}(e, f, g)\\(e, g, f)\\(g, e, f)\\(g, f, e)\\(f, g, e)\\(f, e, g)\end{array}$$

Since there are two original terminal pairs, for each permutation, we view it as a sequence and generate all possible two continuous subsequences. For example, consider (e, f, g). There are four possible ways to partition it into two subsequences:

$$\begin{array}{ll}(\ ) & (e, f, g)\\(e) & (f, g)\\(e, f) & (g)\\(e, f, g) & (\ )\end{array}$$

Each subsequence can be partitioned in this way. After all sequences are partitioned, we assign the first subsequence to (a, b) and the second subsequence to (c, d). Thus from the above pairs of subsequences, we have the following paths:

$$\begin{array}{ll} (a, b) & (c, e, f, g, d) \\ (a, e, b) & (c, f, g, d) \\ (a, e, f, b) & (c, g, d) \\ (a, e, f, g, b) & (c, d) \end{array}$$

It may be wondered why (c, e, d), for instance, is not generated. This will be generated later by applying the procedure to the sequence (f, g, e). (f, g, e) can be partitioned into (f, g) and (e). This will produce the path (c, e, d).

After generating paths, we then scan each path linearly and produce terminal pairs. For instance, for (a, e, b), we have (a, e) and (e, b) as terminal pairs.

Note that given a simple cycle separator, the sets V_a, V_c of nonterminal points of the inputs of the subproblems are all the same. The following procedure is our detailed algorithm to generate all possible input instances of the subproblems:

```
Procedure GEN_INPUTS($V$, $T$, $b$, $\rho$)
Input:  $V = \{v_1, v_2, \ldots, v_n\}$, a set of points;
        $T = \{(t_1, t'_1), (t_2, t'_2), \ldots, (t_m, t'_m)\}$, a set of terminal pairs; and
        $b$, a simple cycle which goes through some points in $V$, $T$, and $I$.
Output: $\rho = \{(V_a, V_c, T_a, T_c) | (V_a, T_a), (V_c, T_c)$ are the candidates of the
            input instances of the interior and exterior subproblems,
            respectively.$\}$

Step 1. $\rho = \varnothing$.
Step 2. Use the simple cycle $b$ to divide the points in $V$ and $T$ into the
        interior part $V_a$, $T'_a$ and the exterior parts $V_c$, $T'_c$, respectively.
        (Apply the point location algorithm of Preparata and Shamos
        (1985).) Let $V_b$ be the nonterminal points of $V$ in the simple cycle.
Step 3. Generate all permutations of points in $V_b$.
Step 4. For each permutation $q' = (v'_1, v'_2, \ldots, v'_k)$, call

                GEN_SEQUENCES ($q'$, $V_s$, 1, 1).

        /* The function of GEN_SEQUENCES views a permutation as
           a sequence and generates all possible sequences of $m$ contin-
           uous subsequences, as defined in the above discussion. */
Step 5. For each $(q_1, q_2, \ldots, q_m) \in V_s$ do:
Step 6.     For each $q_i = (v_{i1}, v_{i2}, \ldots, v_{ik})$, find the set of terminal
            pairs $T_p = \{(t_i, v_{i1}), (v_{i1}, v_{i2}), \ldots, (v_{ik}, t'_i)\}$. If $((t_{ik}, t_{ij}) \in T_p)$ and
            $((t_{ik} \in T'_a, t_{ij} \in T'_c)$ or $(t_{ik} \in T'_c, t_{ij} \in T'_a))$, then jump to step 5 and
            try the next instance.
```

```
Step 7.  Generate $\rho' = \{(V_a, V_c, T_a, T_c) \mid T_a \cup T_c = T_p, T_a \cap T_c = \emptyset$, and if
                  $v_i$ or $v_j \in T'_a$ (resp. $T'_c$), then $(v_i, v_j) \notin T_c$ (resp.
                  $T_a$).$\}$
Step 8.  Let $\rho = \rho' \cup \rho$ and $T_p = \emptyset$.
Step 9. Return $\rho$.
```

Procedure GEN_SEQUENCES(q', V_s, Cur, i)

```
Function: Find $V_s = \{(q_1, q_2, \ldots, q_m) \mid q_i$ is a continuous subsequence of
          $q' = (v'_1, v'_2, \ldots, v'_k)$. If $j1 < j2$, $i1 < i2$ and $v'_{j1} \in q'_{i2}$,
          then $v'_{j2} \notin q'_{i1}$. All points in $q_i$ are disjoint and the
          union of points in $q_i$ is $\{v'_1, v'_2, \ldots, v'_k\}$.$\}$

Step 1. If $i = m$, then $q_i = (v'_{\mathrm{Cur}}, \ldots, v'_k)$ and $V_s = V_s \cup \{(q_1, q_2, \ldots, q_m)\}$
Step 2. Else for $j$ = Cur-1 to $k$ do:
Step 3.   If $j$ = Cur-1, then $q_i = \emptyset$ else $q_i = (v'_{\mathrm{Cur}}, \ldots, v'_j)$ and call
          GEN_SEQUENCES($q'$, $V_s$, $j + 1$, $i + 1$).
```

The time complexity of the recursive procedure GEN_SEQUENCES is equal to the number of patterns generated in the procedure. The number of patterns generated is equal to the number of ways to place k nondistinct objects into m distinct cells. The number of ways to do this is $\binom{m+k-1}{k}$.

Let us examine the time complexity of Procedure GEN_INPUTS. Step 2 takes $(n + 2m) \cdot k$, where k is the number of points in the simple cycle b (Preparata and Shamos, 1985). Step 3 takes $O(k!) \le O(k^k)$. Step 4 takes

$$\binom{m+k-1}{k} \le (m+k)^k.$$

Step 6 takes linear time. Step 7 generates 2^k instances for each $(q_1, q_2, \ldots, q_m) \in V_s$. Therefore the time complexity of this procedure and the number of input instances generated are both bounded by

$$O(((n + 2m) \cdot k + k^k) + (k^k \cdot (m + k)^k) + k^k \cdot (m + k)^k \cdot (k + 2^k)).$$

Because k is the size of the simple cycle,

$$k \le \sqrt{8 \cdot (n + 2m + 3)} \le \sqrt{8 \cdot (n + 2m)} + \sqrt{8 \cdot 3}.$$

We can derive that

$$O(((n + 2m) \cdot k + k^k) + (k^k \cdot (m + k)^k) + k^k \cdot (m + k)^k \cdot (k + 2^k)) \le (n + 2m)^{c_c(\sqrt{n+2m}) + c_d},$$

where c_c and c_d are some constants.

4.4. Algorithm GETSP *and Its Time Complexity.* We have shown how to generate all possible simple cycle separators and how to generate all possible input instances of the two subproblems for a given simple cycle. We now state the algorithm based upon the above two procedures to solve the (V, T)-GETSP problem.

Algorithm GETSP(V, T, E, C)
Input: Two sets $V = \{v_1, v_2, \ldots, v_n\}$ and $T = \{(t_1, t_1'), (t_2, t_2'), \ldots, (t_m, t_m')\}$, where all points are on the plane.
Output: A set E of m paths, where the ith path starts from t_i and terminates at t_i' $(1 \le i \le m)$ and each point in V is passed through by exactly one path, such that total length of all paths is minimized.

```
Step 1.  $C := \infty$.
Step 2.  If $V = \emptyset$, do step 3, else do steps 4–10:
Step 3.    Let $E = \{e \mid e = \overline{t_i t_i'}$, where $1 \le i \le m\}$ and let $C$ be the total
           length of the edges in $E$. Return $E$ and $C$.
Step 4.  Call GEN_CYCLES_B($V, T, B'$).
Step 5.  For each $b \in B'$, do:
Step 6.    Call GEN_INPUTS($V, T, b, \rho$)
Step 7.    For each $(V_a, V_c, T_a, T_c) \in \rho$ do:
Step 8.      Call GETSP($V_a, T_a, E_a, C_a$).
Step 9.      Call GETSP($V_c, T_c, E_c, C_c$).
Step 10.     If $C > C_a + C_c$, then $E = E_a \cup E_c$, $C = C_a + C_c$.
```

We recursively divide these subproblems until $V = \emptyset$. When $V = \emptyset$ it means that there is no nonterminal points. Hence we can directly connect each terminal pair in T and return the total distance. Each input instance derives a solution. We select the instance with the smallest cost as the desired one, and eliminate all other incorrect guesses.

We know that the time complexity and the number of cycles generated in Procedure GEN_CYCLES_B are both bounded by $O((n + 2m)^{c_a(\sqrt{n+2m}) + c_b})$. The time complexity and the number of input instances generated in Procedure GEN_INPUTS are bounded by $O((n + 2m)^{c_c(\sqrt{n+2m}) + c_d})$. Therefore steps 4–6 take $O((n + 2m)^{c_a(\sqrt{n+2m}) + c_b}) \cdot O((n + 2m)^{c_c \cdot (\sqrt{n+2m}) + c_d}) = O((n + 2m)^{c_e \cdot (\sqrt{n+2m}) + c_f})$, where $c_e = c_a + c_c$ and $c_f = c_b + c_d$. Let us consider the size of the subproblems. We know that $|V_a \cup T_a'| \le 2(n + 2m)/3$ and $|V_c \cup T_c'| \le 2(n + 2m)/3$. However, after calling Procedure GEN_INPUTS, all points in the simple cycle become the terminal points in the subproblems. Therefore the sum of numbers of points in V_a and T_a (also in V_c and T_c) is no more than $2(n + 2m)/3 + \sqrt{8 \cdot (n + 2m + 3)}$, for the number of points in the simple cycle separator is no more than $\sqrt{8 \cdot (n + 2m + 3)}$. Let $T(n + 2m)$ be the time complexity of Algorithm GETSP with V and T as inputs, where n is the number of points in V and $2m$ is the number of points in T. Then we have the following formula:

$$T(n + 2m) = O((n + 2m)^{c_e \cdot (\sqrt{n+2m}) + c_f}) \cdot T(2(n + 2m)/3 + \sqrt{8 \cdot (n + 2m + 3)}).$$

Let $n_1 = n + 2m$. Then we have

$$T(n_1) = O(n_1^{c_e \cdot (\sqrt{n_1}) + c_f}) \cdot T(2 \cdot n_1/3 + \sqrt{8 \cdot (n_1 + 3)}).$$

When n_1 is large enough, $n_1/6 \geq \sqrt{8 \cdot (n_1 + 3)}$,

$$T(n_1) = O(n_1^{c_e \cdot (\sqrt{n_1}) + c_f}) \cdot T(5 \cdot n_1/6),$$

$$T(n_1) = O(n_1^{c_e \cdot ((1/(1-5/6)) \cdot \sqrt{n_1}) + \log_{6/5} n_1 \cdot c_f}),$$

$$T(n_1) = O(n_1^{O(\sqrt{n_1})}).$$

Therefore we can derive that

$$T(n + 2m) = O((n + 2m)^{O(\sqrt{n+2m})}).$$

Because the ETSP problem is a special case of the GETSP problem, there is only one terminal pair in T, and the time complexity of solving the ETSP problem is $O(n^{O(\sqrt{n})})$.

5. Concluding Remarks.

The divide-and-conquer strategy is a well-known approach to designing efficient algorithms. In this paper we propose a searching over separators strategy to solve three geometry problems which cannot be solved by the divide-and-conquer strategy directly. All these problems have a common property in their optimal solutions, that is, there exists a separator that can partition the input data into two parts called A and C, such that if we merge the optimal solutions to the two subproblems with A and C as inputs, independently, then we have the optimal solution to the original problem.

Because of the above property, for each case we find a procedure which can generate all possible separators efficiently. We use these possible separators to divide the input into two parts and then recursively solve the subproblems. If the separator is the desired one, then we can find a feasible solution with minimal cost; otherwise we try the next possible separator.

The four geometric NP-hard problems we solve in this paper are the discrete Euclidean P-median problem, the discrete Euclidean P-center problem, the Euclidean P-center problem, and the Euclidean traveling salesperson problem. The time complexity for the DEPM, the DEPC, and the EPC problems is $O(n^{O(\sqrt{P})})$ and for the ETSP problem it is $O(n^{O(\sqrt{n})})$. The best previous results are $O(n^{P+1} \cdot P)$ (Papadimitriou, 1981) for the DEPM problem, $O(n^{2P-1} \cdot \log n)$ (Drezner, 1984; Megiddo, 1983) for the EPC problem, and $O(n^2 2^n)$ (Held and Karp, 1962; Horowitz and Sahni, 1978; Lawler *et al.*, 1985) for the ETSP problem.

We strongly recommend the reader to consult Smith (1991) which contains many ideas similar to ours. Smith used his strategy to solve the Steiner tree problem and the traveling salesperson problem in subexponential time.

References

Aho, A. V., Hopcroft, J. E., and Ullman, J. D., *The Design and Analysis of Computer Algorithms*, Bell Telephone Laboratories, Inc., New York, 1976.

Bentley, J. L., Divide and Conquer Algorithms for Closest Point Problems in Multidimensional Space, Ph.D. Thesis, Department of Computer Science, University of North Carolina, 1976.

Bentley, J. L., Multidimensional Divide-and-Conquer, *Communications of the Association for Computing Machinery*, Vol. 23, 1980, pp. 214–229.

Delaunay, B., Sur la sphère vide, *Izvestiya Akademii Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk*, Vol. 7, 1934, pp. 793–800.

Drezner, Z., The *P*-Center Problem—Heuristics and Optimal Algorithms, *Journal of the Operational Research Society*, Vol. 35, No. 8, 1984, pp. 741–748.

Drezner, Z., On the Rectangular *P*-Center Problem, *Naval Research Logistics*, Vol. 34, 1987, pp. 229–234.

Edelsbrunner, H., *Algorithms in Combinatorial Geometry*, Springer-Verlag, New York, 1987.

Held, M., and Karp, R. M., A Dynamic Programming Approach to Sequencing Problems, *SIAM Journal on Applied Mathematics*, Vol. 10, 1962, pp. 196–210.

Horowitz, E., and Sahni, S., *Fundamentals of Computer Algorithms*, Computer Science Press, Rockville, MD, 1978.

Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G., and Shmoys, D. B., *The Traveling Salesman Problem—A Guided Tour of Combinatorial Optimization*, Wiley–Interscience, New York, 1985.

Lipton, R., and Tarjan, R. E., A Separator Theorem for Planar Graphs, *SIAM Journal on Applied Mathematics*, Vol. 36, No. 2, 1979, pp. 177–189.

Megiddo, N., Linear-Time Algorithms for Linear Programming in R^3 and Related Problems, *SIAM Journal on Computing*, Vol. 12, No. 4, 1983, pp. 759–776.

Megiddo, N., and Supowit, K. J., On the Complexity of Some Common Geometric Location Problems, *SIAM Journal on Computing*, Vol. 13, No. 1, 1984, pp. 182–196.

Mehlhorn, K., *Data Structure and Algorithms 3: Multi-dimensional Search and Computational Geometry*, Springer-Verlag, Berlin, 1984.

Miller, G. L., Finding Small Simple Cycle Separators for 2-Connected Planar Graphs, *Journal of Computer and System Sciences*, Vol. 32, 1986, pp. 265–279.

Nishizeki, T., and Chiba, N., *Planar Graphs, Theory and Algorithms*, Elsevier, Amsterdam, 1988.

Papadimitriou, C. H., Some Computational Problems Related to Database Concurrent Control, *Proceedings of the Conference on Theoretical Computer Science*, University of Waterloo, Waterloo, Ontario, 1977, pp. 275–282.

Papadimitriou, C. H., Worst-Case and Probabilistic Analysis of a Geometric Location Problem, *SIAM Journal on Computing*, Vol. 10, 1981, pp. 542–557.

Papadimitriou, C. H., and Steiglitz, K., Some Complexity Results for the Traveling Salesman Problem, *Proceedings of the 8th Annual ACM Symposium on Theory of Computing*, New York, 1976, pp. 1–9.

Preparata, F. P., and Shamos M. I., *Computational Geometry*, Springer-Verlag, New York, 1985.

Smith, W. D., Studies in Computational Geometry Motivated by Mesh Generation, accepted by *Algorithmica*, 1991.

Voronoi, G., Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Premier Mémoire: Sur quelques propriétés des formes quadratiques positives parfaites, *Journal für die Reine und Angewandte Mathematik*, Vol. 133, 1907, pp. 97–178.