MATHEMATICS OF OPERATIONS RESEARCH
Vol. 2, No. 3, August 1977
Printed in U.S.A.

PROBABILISTIC ANALYSIS OF PARTITIONING ALGORITHMS FOR THE TRAVELING-SALESMAN PROBLEM IN THE PLANE*†

RICHARD M. KARP

University of California, Berkeley

We consider partitioning algorithms for the approximate solution of large instances of the traveling-salesman problem in the plane. These algorithms subdivide the set of cities into small groups, construct an optimum tour through each group, and then patch the subtours together to form a tour through all the cities. If the number of cities in the problem is n, and the number of cities in each group is t, then the worst-case error is $O(\sqrt{n/t})$. If the cities are randomly distributed, then the relative error is $O(t^{-1/2})$ (with probability one). Hybrid schemes are suggested, in which partitioning is used in conjunction with existing heuristic algorithms. These hybrid schemes may be expected to give near-optimum solutions to problems with thousands of cities.

1. Introduction. By the traveling-salesman problem in the plane we mean the problem of constructing a polygon of minimum perimeter through a given set of points (cities) in the plane. There has been considerable investigation of heuristic methods for the solution of this problem. Computer programs based on local improvement techniques [13] or other heuristic principles [11] appear to give near-optimal solutions to problem instances with two or three hundred cities, without using excessive amounts of computer time. Good results have also been obtained using man-machine systems, in which a person, communicating with a computer through a display terminal, controls the search for a solution [1], [12], [14]. Success on problems of modest size has also been achieved by persons armed with pegs to mark the cities and string to measure distances [5].

On the other hand, at the present state of the art it is quite impossible to find, and prove that one has found, the strictly optimal solution to a large problem. The most effective exact solution methods are based on branch-and-bound techniques [7], [9], [10], [19]; they solve 60-city problems routinely, but use excessive amounts of computer time on problems with one hundred cities. The fact that the traveling-salesman problem in the plane is NP-hard [6], [15] provides convincing evidence that there does not exist a polynomial-time algorithm capable of solving the problem exactly.

Recently attention has turned to the construction of polynomial-time algorithms guaranteed to solve the problem within a specified approximation [4], [16]. The best result along these lines is due to Christofides [4], who has given an algorithm that runs in time $O(n^3)$, and always yields a tour less than 50% longer than the optimum tour.

The present paper takes a probabilistic approach. We assume that the cities are scattered at random in a rectangular region X of the plane. We exhibit a family of algorithms with the following property: for every $\epsilon > 0$ there is an algorithm $\mathcal{A}(\epsilon)$ in the family such that (a) $\mathcal{A}(\epsilon)$ runs in time $C(\epsilon)n + O(n \log n)$; (b) with probability 1, $\mathcal{A}(\epsilon)$ produces a tour costing not more than $(1 + \epsilon)$ times the cost of an optimal tour.

* Received May 18, 1977; revised July 15, 1977.
AMS 1970 subject classification. Primary 90C10. Secondary 90C35.
IAOR 1973 subject classification. Main: Network Programming. Cross Reference: Computational Analysis.
Key words: Euclidean traveling-salesman problem, approximation algorithm, geometric probability, heuristic algorithm, partitioning.
† Research supported by National Science Foundation Grant MCS74-17680-A02.

The algorithms are based on partitioning the region X into "small" subregions, each of which contains about t cities. An optimum tour is constructed within each subregion, and these subtours are then combined to yield a tour through all the cities. Of course, standard heuristic methods may be used instead of exact solution methods to find the tours through the subregions. Such a combination of partitioning with existing heuristic methods should make it feasible to find near-optimal solutions to problems with many thousands of cities.

2. Tours and spanning walks. The traveling-salesman problem in the plane asks for a polygon of minimum length through a given set of points. Such a polygon corresponds to a closed tour in which each city is visited exactly once. In designing algorithms for the problem it is convenient to allow a larger set of feasible solutions, corresponding to tours which visit some cities repeatedly. This short section is devoted to showing that such a change in the problem statement makes no real difference.

Let V be a set of points in the plane. For $u \in V$ and $v \in V$, let $d(u, v)$ be the Euclidean distance between u and v. Given any multigraph (possibly with loops or multiple edges) $G = (V, E)$, with vertex set V and edge set E, define $w(G)$, the *weight* of G, as $\sum_{\{u, v\} \in E} d(u, v)$; here $d(u, v)$ is counted multiply if $\{u, v\}$ is a multiple edge. The graph $G = (V, E)$ is a *tour* if G is connected and every vertex has degree 2; G is a *spanning walk* if G is connected and all vertices are of even degree (a loop at v contributes 2 to the degree).

LEMMA 1. *Let G be a spanning walk. Then there is a tour H such that $w(H) \leqslant w(G)$.*

PROOF. We define two operations on a multigraph $G = (V, E)$ at a vertex v. (a) If there is a loop v then the operation LOOP(v) is applicable; it deletes the loop. (b) If $\{u, v\} \in E$, $\{w, v\} \in E$ and the pair of arcs $\{\{u, v\}, \{w, v\}\}$ is not a cut set of G, then the operation PASS(u, v, w) is applicable; it deletes the arcs $\{u, v\}$ and $\{w, v\}$, and adds the arc $\{u, w\}$. We claim (omitting the easy proof) that: (i) the application of any operation transforms G to another spanning walk G'; (ii) $w(G') \leqslant w(G)$; this follows from the triangle inequality $d(u, v) + d(v, w) \geqslant d(u, w)$; and (iii) if v is of degree > 2 in G, then some operation at v is applicable.

Repeated application of operations yields the desired tour H.

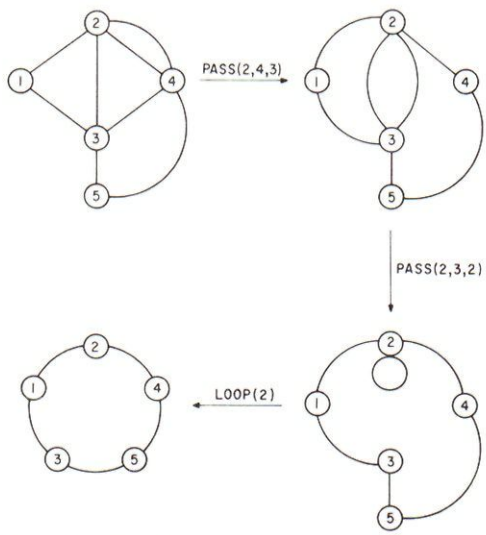

FIGURE 1. Transforming a Spanning Walk to a Tour.

3. A partitioning algorithm. In this section we present a partitioning algorithm (called Algorithm 1) for the construction of a spanning walk through n given points (*cities*) in a rectangular region of the plane. The algorithm uses a subroutine TOUR capable of the exact solution of t-city traveling-salesman problems, where t is specified by the user of the algorithm. We show that the execution time of Algorithm 1 is $O(n \log n)$ plus the time for $(n-1)/(t-1)$ calls on TOUR, and that $|W_1| - |T^*| = O(\sqrt{n/t})$, where $|W_1|$ is the length of the spanning walk W_1 produced by the algorithm, and $|T^*|$ is the length of an optimum tour T^*. It follows that, if the cities are distributed at random then, with probability tending to 1, $|W_1|/|T^*| = 1 + O(t^{-1/2})$.

Specification of Algorithm 1

Let n be the number of cities, let t be a parameter which will serve as an upper bound on the sizes of subproblems solved exactly by the subroutine TOUR, and let $k(n) = \lceil \log_2(n-1)/(t-1) \rceil$; when n is clear from context we write k instead of $k(n)$.

Algorithm 1 proceeds by subdividing the original rectangle into 2^k subrectangles, each containing at most t of the cities. Subroutine TOUR is then called to construct an optimum tour through the cities in each subrectangle. The subdivision is such that the union of the 2^k subtours forms a spanning walk through all n cities. The operations LOOP and PASS introduced in Lemma 1 may then be used to transform this walk to a tour.

We assume for convenience that no two cities are at exactly the same distance from any side of the original rectangle.

Let Y be a rectangle containing m cities. Assume Y is placed so that its longer side is horizontal. Let x be the $\lceil m/2 \rceil$th closest city to the left edge of Y. A vertical cut through x subdivides Y into a "left rectangle" $l(Y)$ and a "right rectangle" $r(Y)$, having x on their common boundary. The construction is indicated in Figure 2.

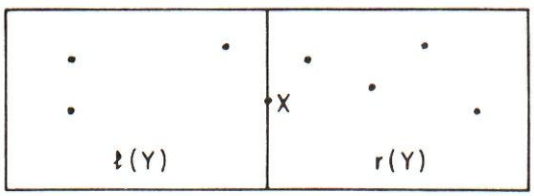

FIGURE 2. Partitioning a Rectangle by a Cut in the Shorter Direction.

Note that a spanning walk through the cities in $l(Y)$, plus a spanning walk through the cities in $r(Y)$, constitutes a spanning walk through the cities in Y.

Now we are ready to define our algorithm. Procedure A1 takes a rectangle X as input and produces as output a spanning walk through the cities in X. The quantity $n(X)$ denotes the number of cities in X. In the course of the definition a recursive procedure WALK occurs. This procedure takes as inputs a rectangle Y and a nonnegative integer j. The output of WALK is a spanning walk through the cities in Y. The argument j determines the depth of the recursion used in constructing this walk. At the base of the recursion ($j = 0$), WALK calls on a subroutine TOUR(Y) that constructs an optimum tour through the cities in Y.

PROCEDURE A1
A1(X) = WALK(X, $k(n(X))$)
WALK(Y, j) = *if* $j = 0$
then TOUR(Y)
else WALK($l(Y), j-1$) $\cup$ WALK($r(Y), j-1$).

Figure 3 shows the result of applying Algorithm 1 to an example with $n = 25$, $t = 4$ and $k = 3$. The walk is the union of eight quadrilaterals. Figure 4 gives a tour obtained from this walk by the technique of Lemma 1.

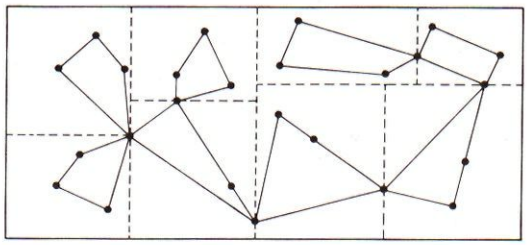

FIGURE 3. Walk Created by Algorithm 1.

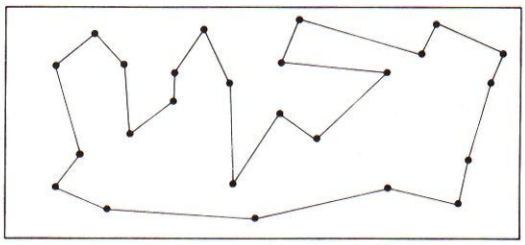

FIGURE 4. Tour Obtained Using the Loop and Pass Operations.

Correctness of Algorithm 1

LEMMA 2. *The result of applying* A1(X) *is a spanning walk through the cities in* X. *Each time* TOUR(Y) *is called,* Y *contains at most* t *cities.*

PROOF. Induction on k shows that WALK(Y, j) delivers a spanning walk through the cities in X; the first statement follows. A second induction, through decreasing values of j, shows that, whenever WALK(Y, j) is called, the number of cities in Y is $\leqslant 2^j(t-1)+1$; since TOUR(Y) is called by WALK(Y, j) only when $j = 0$, the second result follows.

Analysis of the execution time of Algorithm 1

In the following analysis we assume that Algorithm 1 is to be implemented on a random-access computer that requires one unit of time to compare or add real numbers (such as the x- or y-coordinates of two cities). We assume there are constants D and d such that TOUR( ) requires time $\leqslant Dd^t$ to solve a t-city problem. For example, if the standard dynamic programming algorithm with execution time $t^2 \cdot 2^t$ is used [3], [8], then any value > 2 can be used for d.

THEOREM 1. *With suitable implementation, Algorithm* 1 *operates within the time bound*

$$2^{k(n)}Dd^t + O(n \log n) < 2\frac{n-1}{t-1}Dd^t + O(n \log n).$$

PROOF. The term $2^{k(n)}Dd^t$ bounds the total time spent in executing procedure TOUR (i.e., solving small traveling-salesman problems).

The remaining work is dominated by the computations of $l(Y)$ and $r(Y)$. Assume inductively that when we are ready to compute $l(Y)$ and $r(Y)$, we have available $n(Y)$, the number of cities in Y, as well as linked lists $H(Y)$ and $V(Y)$; $H(Y)$ contains the cities in Y listed in left-to-right order, and $V(Y)$ contains these cities

listed in bottom-to-top order. Setting up these lists initially requires sorting the n cities on their horizontal and vertical coordinates, which can be done in $O(n \log n)$ steps. Thereafter we can process each Y in time proportional to $n(Y)$, producing $l(Y)$, $r(Y)$, $n(l(Y))$, $H(l(Y))$, $V(l(Y))$, $n(r(Y))$, $H(r(Y))$ and $V(r(Y))$ as output. The total work for these computations is $O(n \log n)$ for the initial sorting, and $O(nk) = O(n \log n)$ for the subsequent processing.

A cutting game

Our next objective is to derive an upper bound on the difference between the cost of the walk produced by our algorithm and the cost of an optimum tour.

In preparation for this analysis we introduce a game involving the subdivision of a rectangle X into subrectangles. There are two players, called Min and Max. The play requires k rounds. Each round consists of a move by Min, and then a move by Max. At the beginning of round l, X has been subdivided into 2^{l-1} subrectangles $\{X_i\}$. During round l, each of the X_i is cut in two, by either a vertical or a horizontal cut. Min's move consists of deciding, independently for each X_i, whether the cut dividing X_i will be vertical or horizontal. Max then chooses the location of the cut. At the end of the k rounds of play, Min pays Max an amount equal to the sum of the perimeters of the 2^k rectangle produced in round k.

Figure 5 shows a play of the 3-round game. Each cut is labelled with the number of the round in which it is played.

By the *short strategy* for Min we mean the policy of choosing, for each rectangle which occurs, the direction parallel to the shorter side. By the *bisection strategy* for Max we mean the policy of placing each cut so as to divide a rectangle into equal halves.

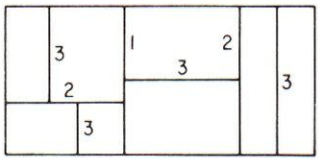

FIGURE 5. A Play of the Cutting Game.

THEOREM 2. *The short strategy is optimal for* Min *and the bisection strategy is optimal for* Max.

PROOF. First we show that the short strategy is best against the bisection strategy. However Min plays against the bisection strategy, the result of the play will be a subdivision of X into 2^k rectangles of equal area. If X is $a \times b$, then each of these rectangles will be $2^{-l}a \times 2^{-m}b$, where l and m are nonnegative integers adding to k. Since the perimeter of a rectangle of fixed area is an increasing function of the longer side, the most favorable choice of l and m for Min is the one that minimizes $\text{Max}\{2^{-l}a, 2^{-m}b\}$. The short strategy achieves this optimal choice simultaneously for all 2^k rectangles occurring in the subdivision.

Next we show by induction on k that the bisection strategy is best for Max against the short strategy. This is certainly true for $k = 1$, where any strategy for Max is best against the short strategy. Assume it as an induction hypothesis for $k = l$. Since we know already that the short strategy is best against the bisection strategy, we can conclude that the short strategy and the bisection strategy form an optimal strategy pair for any l-round game. Now consider an $(l + 1)$-round game on a $a \times b$ rectangle, with $a < b$. Suppose Min, following the short strategy, specifies a cut parallel to the short side. If Max bisects we get two $a \times b/2$ rectangles, and optimal play (with Min using the short strategy and Max using the bisection strategy) for the l ensuing rounds

yields 2^{l+1} congruent rectangles of size, say, $\alpha a \times \beta b/2$, where $\alpha\beta = 2^{-l}$. The value of the game to Max is

$$2^{l+1}(2\alpha a + 2\beta b/2) = 2^{l+2}\alpha a + 2^{l+1}\beta b.$$

On the other hand, if Max does not bisect we get an $a \times b_1$ rectangle and an $a \times (b - b_1)$ rectangle, with $b_1 \neq b/2$. Suppose that, in the ensuing play, Max uses the bisection strategy (which is known to be optimal), but Min, possibly deviating from optimal play, chooses the same directions he would have chosen if Max had bisected in the first round. Then we get 2^l $\alpha a \times \beta b_1$ rectangles, and 2^l $\alpha a \times \beta(b - b_1)$ rectangles, for a total payoff of

$$2^l(2\alpha a + 2\beta b_1) + 2^l(2\alpha a + 2\beta(b - b_1)) = 2^{l+2}\alpha a + 2^{l+1}\beta b.$$

Thus, if Max fails to bisect in the first round, Min can achieve at least as much as he could have achieved if Max had bisected. It follows that bisection is best for Max against the short strategy in the $(l + 1)$-round game, and the induction step is complete.

Finally, since the short strategy and the bisection strategy are best against each other, they form a saddle point, or optimal pair of pure strategies, for the cutting game.

Let $F_k(a, b)$ denote the value (to Max) of a k-round cutting game on an $a \times b$ rectangle.

COROLLARY 1. (a)

$$F_k(a, b) = \min_{\substack{s \text{ integer} \\ s+t=k}} 2(2^t a + 2^s b)$$

(b) *If a and b are held fixed, then* $\sup_k(F_k(a, b))/2^{k/2}$ *exists.*

Error analysis of Algorithm 1

We are now ready to apply our results about the cutting game in an error analysis of Algorithm 1. First we establish notation. Let per(Y) denote the perimeter of rectangle Y, and let $|W|$ denote the length of the walk W (i.e., the sum of the lengths of the occurrences of line segments in W). Let X be an $a \times b$ rectangle containing n cities. Let T^* denote an optimum tour through the n cities, let W_1 denote the walk produced by algorithm 1, and let $k = k(n) = \lceil \log_2(n - 1)/(t - 1) \rceil$.

THEOREM 3. *Let Y be a rectangle within X. Let $T(Y)$ be an optimum tour through the cities in Y. Then* $|T(Y)| - |T^* \cap Y| \leqslant (3/2)\text{per}(Y)$.

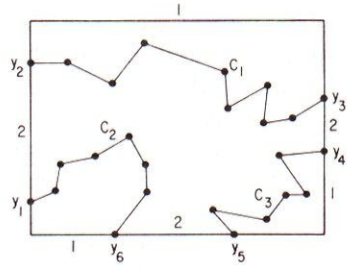

FIGURE 6. Converting $T^* \cap Y$ to a Walk $W(Y)$.

PROOF. Let $T^* \cap Y$ consist of k continuous curves $C_1, C_2, \ldots, C_k$. Let the $2k$ end points of these curves, in clockwise order around the boundary of Y, be y_1,

$y_2, \ldots, y_{2k}$. Assume without loss of generality that $\overline{y_1y_2} + \overline{y_3y_4} + \cdots + \overline{y_{2k-1,2k}} \leqslant \overline{y_2y_3} + \overline{y_4y_5} + \cdots + \overline{y_{2k,1}}$, where $\overline{y_iy_j}$ denotes the distance from y_i to y_j along the perimeter of Y. Consider the walk $W(Y)$ consisting of the following three parts: the curves $C_1, C_2, \ldots, C_k$; two copies of each of the segments y_1y_2, $y_3y_4, \ldots, y_{2k-1,2k}$; plus one copy of each of the segments $y_2y_3, y_4y_5, \ldots, y_{2k,1}$. Then the length of the first part is $|T^* \cap Y|$, and the sum of the lengths of the second and third parts is less than or equal to $3/2$ the perimeter of Y. Thus

$$|T(Y)| \leqslant |W(Y)| \leqslant |T^* \cap Y| + \tfrac{3}{2}\text{per}(Y).$$

Figure 7 indicates a family of examples for which $|T(Y)| - |T^* \cap Y|$ approaches $(3/2)$ per (Y). Let Y be an $\epsilon \times 1$ rectangle, where ϵ is small; let the cities occur more and more densely on the dotted line segments; and let $T^* \cap Y$ be as indicated in Figure 7b. Then $|T^* \cap Y| \to 1$, $|T(Y)| \to 4 + 2\epsilon$, and $|T(Y)| - |T^* \cap Y| \to 3 + \epsilon \to ((3 + \epsilon)/(1 + \epsilon)) \cdot \frac{1}{2}\text{per}(Y)$.

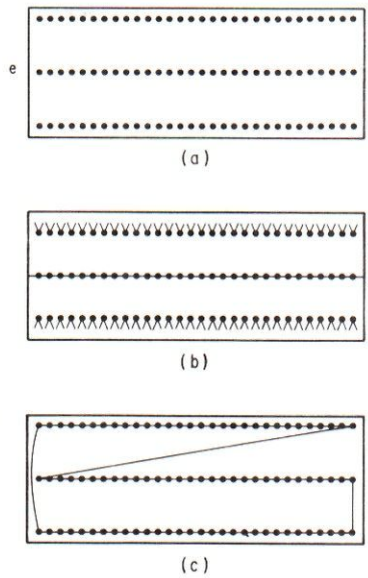

FIGURE 7. An Adverse Distribution of Points.
(a) Points in a Rectangle.
(b) $T^* \cap Y$.
(c) $T(Y)$.

THEOREM 4. $|W_1| - |T^*| \leqslant (3/2)F_k(a, b)$.

PROOF. The execution of Algorithm 1 subdivides X into 2^k subrectangles, $\{Y_i\}$, $i = 1, 2, \ldots, 2^k$, and may be regarded as a play of a k-round cutting game on X. Since every cut is parallel to the short side of its rectangle, the play may be regarded as one in which Min uses the short strategy, which is optimal. Thus $\sum_{i=1}^{2^k} \text{per}(Y_i) \leqslant F_k(a, b)$. But $|W_1| = \sum_{i=1}^{2^k} |T(Y_i)|$ and, applying Theorem 3,

$$\sum_{i=1}^{2^k} |T(Y_i)| \leqslant \sum_{i=1}^{2^k} |T^* \cap Y_i| + \tfrac{3}{2}\text{per}(Y_i) \leqslant |T^*| + \tfrac{3}{2}\sum_{i=1}^{2^k} \text{per}(Y_i) \leqslant |T^*| + \tfrac{3}{2}F_k(a, b).$$

Now regard a and b as fixed and n and t as variable. Then the error bound $(3/2)F_k(a, b) \sim 2^{k/2} \sim \sqrt{n/t}$. Thus, we have

COROLLARY 2. $|W_1| - |T^*| = O(\sqrt{n/t})$.

The following construction, which we sketch informally, shows that the growth rate of our error estimate cannot be improved. Let X be the unit square, let $k(n)$ be even and let t be a multiple of 4. Subdivide X into 2^k congruent subsquares Y_i, $i = 1, 2, \ldots, 2^k$, each of side $2^{-k/2}$. Then it is possible to place the cities such that Algorithm 1 produces the subdivision $\{Y_i\}$, and such that the t cities in each subsquare Y_i fall into four clusters of size $t/4$, with one cluster infinitesimally close to each corner of Y_i. Then

$$F_k(1, 1) = 4 \cdot 2^{k/2}, \qquad \sum_{i=1}^{2^k} |T(Y_i)| \sim 2^k(4 \cdot 2^{-k/2}) = 4 \cdot 2^{k/2},$$

$$|T^*| \sim (2^{k/2} + 1)^2 \cdot 2^{-k/2} \sim 2^{k/2},$$

and

$$\sum_{i=1}^{2^k} |T(Y_i)| - |T^*| \sim 3 \cdot 2^{k/2} = \tfrac{3}{4} F_k(1, 1).$$

4. Random traveling-salesman problems in the plane. In this section we discuss a theorem due to Beardwood, Halton and Hammersley [2], showing that, if the cities are randomly distributed in a region of the plane, then the length of the shortest tour tends to grow as the square root of the number of cities. Since Algorithm 1 produces a spanning walk whose cost differs from the cost of an optimum tour by $O(\sqrt{n/t})$, it follows that the ratio of the cost of this walk to the cost of an optimum tour tends to vary with the parameter t as $1 + O(t^{-1/2})$.

We model a random distribution of points in a region X of the plane by a two-dimensional Poisson distribution $\Pi_n(X)$. The distribution $\Pi_n(X)$ is determined by the following assumptions:

(i) the numbers of cities occurring in two or more disjoint subregions are distributed independently of each other;

(ii) the expected number of cities in a region A is $nv(A)$, where $v(A)$ is the area of A; and

(iii) as $v(A)$ tends to zero, the probability of more than one city occurring in A tends to zero faster than $v(A)$.

From these assumptions it follows that

(iv) $\Pr\{A \text{ contains exactly } m \text{ cities}\} = e^{-\lambda}\lambda^m/m!$, where $\lambda = nv(A)$.

We study the random variable $T_n(X)$, which denotes the length of a shortest tour through the cities in X, assuming that the set of cities is distributed according to $\Pi_n(X)$.

THEOREM 5 [2]. *There exists a positive constant β (independent of X) such that $T_n/\sqrt{nv(X)} \to \beta$ with probability* 1.

The technical meaning of this statement is as follows. Suppose we form an infinite sequence $Z_1, Z_2, \ldots, Z_n, \ldots$ of independent samples, where Z_n is drawn from the distribution of $T_n/\sqrt{nv(X)}$. Then, for every $\epsilon > 0$, $\Pr\{\lim|Z_n - \beta| > \epsilon\} = 0$.

Thus, when n is sufficiently large, one can predict the value of $T_n/\sqrt{n}$ closely with a high probability of being correct. The result of Beardwood, Halton and Hammersley applies not only to rectangles, but to all Lebesgue measurable regions; replacing $\sqrt{n}$ by $n^{(d-1)/d}$, their result also applies to traveling-salesman problems in Euclidean d-space.

Combining Theorem 5 with our analysis of Algorithm 1, we can state the following result, which establishes that Algorithm 1 yields a "probabilistic ϵ-approximation scheme" for the solution of the traveling-salesman problem in the plane.

THEOREM 6. *There are constants D_1 and d_1 such that, for every $\epsilon > 0$, we can construct an algorithm $\mathcal{A}_1(\epsilon)$ with the following properties*: (i) *$\mathcal{A}_1(\epsilon)$ runs within time $D_1\epsilon^2 d_1^{1/\epsilon^2} n + O(n \log n)$; and* (ii) *with probability* 1, *$\mathcal{A}(\epsilon)$ constructs a tour of length $< (1 + \epsilon)$ times the cost of an optimum tour.*

PROOF. Corollary 2 tells us that $|W_1| - |T^*| < C\sqrt{n/t}$. Thus, the relative error is $< C't^{-1/2}/\beta$, with probability 1, for any $C' > C$; $\mathcal{A}(\epsilon)$ is simply Algorithm 1, with $t > C^2/\beta^2\epsilon^2$. By Theorem 1, the running time is $< 2(n-1)Dd^t/(t-1) + O(n \log n)$. The result follows if we set $D_1 = 2\beta^2/C^2$, and $d_1 = d^{C^2/\beta^2}$.

We shall be interested in the expected performance of a partitioning algorithm for the traveling-salesman problem in the plane.

This leads us to investigate the quantity $\beta_X(t) = E(T_t(X))/\sqrt{t}$. By a construction given in [2] there exists a constant C depending on X such that the length of a shortest tour through any n points in X is $\leqslant C\sqrt{n}$. From this it follows that $\beta_X(t)$ is uniformly bounded. Hence, using Theorem 5 and the dominated convergence theorem [20, p. 206], it follows that $\beta_X(t) \underset{t\to\infty}{\to} \beta\sqrt{v(X)}$. Here we study the rate of convergence.

Let the $a \times b$ rectangle X be fixed throughout the following discussion. Assume that dimensions are scaled so that $ab = 1$.

THEOREM 7. *For all t, $\beta_X(t) - \beta \leqslant 6(a+b)/\sqrt{t}$.*

PROOF. Consider a problem instance drawn for $\Pi_{4t}(X)$. Let T^* be an optimum tour. Subdivide X into four $a/2 \times b/2$ rectangles Y_1, Y_2, Y_3, Y_4. Let $T(Y_i)$ denote the length of a shortest tour through the cities in Y_i. By Theorem 3, $|T(Y_i)| \leqslant |T^* \cap Y_i| + (3/2)\text{per}(Y_i)$. Hence, $\sum_{i=1}^4 |T(Y_i)| \leqslant |T^*| + 6(a+b)$. But $E(|T^*|) = \beta_X(4t)\sqrt{4t}$ and $E(|T(Y_i)|) = \frac{1}{2}\beta_X(t)\sqrt{t}$ since the set of cities in Y_i is distributed as if it were drawn from $\Pi_t(X)$, and then had all dimensions scaled down by a factor of $\frac{1}{2}$. Hence,

$$\beta_X(t) \leqslant \beta_X(4t) + 3(a+b)\cdot\frac{1}{\sqrt{t}}.$$

By induction on k,

$$\beta_X(t) \leqslant \beta_X(4^k t) + 3(a+b)\cdot\frac{1}{\sqrt{t}}\left(1 + \frac{1}{2} + \cdots + \frac{1}{2^{k-1}}\right)$$

$$\leqslant \beta_X(4^k t) + 3(a+b)\cdot\frac{2}{\sqrt{t}}.$$

Since $\beta_X(4^k t) \underset{k\to\infty}{\to} \beta$, the theorem follows.

5. Expected performance of a partitioning algorithm. In this section we assume that the set of cities is drawn from $\Pi_n(X)$. We consider a variant of Algorithm 1 in which the rectangle X is partitioned into subrectangles, in each of which the expected number of cities is t. An optimum tour is constructed in each subrectangle, and these tours are then patched together to form a walk W_2 through all the cities. Our main result is that

$$E(|W_2|)/\sqrt{n} \leqslant \beta_X(t) + O(t^{-7/6}).$$

Specification of Algorithm 2

Throughout the following discussion X is a fixed rectangle of area 1, and t is a fixed positive real number. Choose $k = k(n)$ as the least positive integer such that $t \cdot 2^{k(n)} \geqslant n$. Given any rectangle Y, define $l'(Y)$ and $r'(Y)$ as the two subrectangles determined by a bisecting cut parallel to the short sides of Y. Define $e(Y)$ as the shortest line segment joining a city in $l'(Y)$ with a city in $r'(Y)$; if either $l'(Y)$ or $r'(Y)$ contains no city, then $e(Y)$ is undefined.

In the following procedure definition, procedure A2 takes a rectangle X as input and produces as output a spanning walk through the cities in X. The recursive procedure WALK2(Y, j) accepts as input a rectangle Y and a positive integer j, and produces a spanning walk through the cities in Y. The input j controls the depth of recursion. The procedure TOUR(Y) is a subroutine that constructs an optimum tour through the cities in Y.

PROCEDURE A2
A2(X) = WALK2(X, k)
WALK2(Y, j) = *if* $j = 0$
then TOUR(Y)
else WALK2($l'(Y), j - 1$). $\cup$ WALK2($r'(Y), j - 1$) $\cup$ $2e(Y)$.

Alternately, Algorithm 2 may be viewed as follows. The rectangle X is subdivided into 2^k subrectangles according to a play of the cutting game in which the two players use the short strategy and the bisection strategy, respectively. The spanning walk W_2 consists of shortest tours through these subrectangles, together with additional line segments linking these tours together into a connected structure; each of these connecting segments occurs twice.

Figure 8 shows the result of applying Algorithm 2 to the example of Figure 3. The parameters are $n = 25$, $t = 25/8$, $k = 3$. Each cross-hatched segment in Figure 8 occurs twice in W_2.

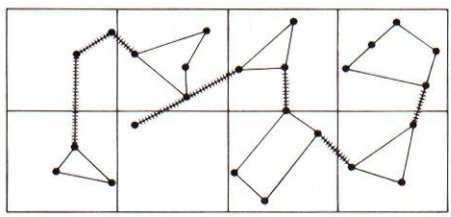

FIGURE 8. Example of the Construction of W_2.

LEMMA 3. *The result of applying Algorithm* 2 *is an Eulerian walk through the cities in* X. *Each time* TOUR(Y) *is called, the expected number of cities in* Y *is* $n/2^k \leqslant t$.

The expected execution time of Algorithm 2

In analyzing the performance of Algorithm 2, we make the following assumption about the subroutine TOUR.

ASSUMPTION. *Let* $E(s, Y)$ *denote the expected time for* TOUR *to compute a shortest tour through* s *points randomly distributed in the rectangle* Y. *Then there are absolute constants* c *and* C *such that, for all* s *and* Y, $E(s, Y) \leqslant Cc^s$.

If TOUR is the standard dynamic programming alorithm [3], [8] with execution time $O(s^2 \cdot 2^s)$ then any constant $c > 2$ will work. Certain branch-and-bound methods

[7], [9], [10], [19] seem to achieve substantially smaller values of c, but no rigorous analyses exist.

THEOREM 8. *Let c and C be as in the Assumption. Then, with suitable implementation, the expected execution time of Algorithm 2 on problems drawn from $\Pi_n(X)$ is less than or equal to $2C(n/t)e^{(c-1)t} + O(n \log^2 n)$.*

PROOF. The execution time is the sum of three contributions: (i) the time spent solving "small" traveling-salesman problems using the subroutine TOUR; (ii) the time spent determining the line segments $e(Y)$; and (iii) the time spent on other operations. Contribution (iii) can be bounded by $O(n \log n)$, exactly as in the analysis of Algorithm 1.

We estimate contribution (i) as follows. The subroutine TOUR is invoked $2^{k(n)}$ times. Each time the number of cities has a Poisson distribution with mean t. Thus the expected execution time of each invocation is $\leqslant C\sum_{x=0}^{\infty} e^{-t}(t^x/x!)c^x = Ce^{(c-1)t}$. The expected total time spent in TOUR is thus $\leqslant 2^{k(n)}Ce^{(c-1)t} < (2n/t)Ce^{(c-1)t}$.

Finally, we show that contribution (ii) is $O(n \log^2 n)$. As a first step, we show that the time to compute $e(Y)$, the shortest segment joining a city in $l'(Y)$ with a city in $r'(Y)$, is $O(n(Y)\log n(Y))$. Each candidate for $e(Y)$ must cross B, the cut separating $l'(Y)$ from $r'(Y)$. For each city $a \in l'(Y)$, define the interval $I(a)$ by:

$$I(a) = \{x \in B \mid a \quad \text{is the closest city in} \quad l'(Y) \quad \text{to} \quad x\};$$

similarly, for any city $b \in r'(Y)$,

$$J(b) = \{x \in B \mid b \quad \text{is the closest city in} \quad r'(Y) \quad \text{to} \quad x\}.$$

Using techniques due to Shamos and Hoey [17], [18] these intervals can be determined in $O(n(Y)\log n(Y))$ steps. Then, in linear time, one can list all pairs a, b such that $I(a) \cap J(b)$ has positive measure. There are at most $n(Y)$ such pairs, and they are the only candidates for the segment $e(Y)$. Thus $e(Y)$ can be determined in $O(n(Y) \log n(Y))$ steps. The expected time spent in computing the segments $e(Y)$ thus grows as

$$E\left(\sum_{\{Y \mid Y \text{ is subdivided}\}} n(Y)\log n(Y)\right).$$

Application of Chebyshev's inequality yields the result that, if $n(Y)$ is Poisson distributed with mean λ, then $E(n(Y)\log_2 n(Y)) \leqslant \lambda \log_2 \lambda + 4/3$. Hence,

$$\begin{aligned} E\left(\sum_{\{Y \mid Y \text{ is subdivided}\}} n(Y)\log n(Y)\right) &\leqslant \sum_{j=0}^{k-1} 2^j\left(\frac{n}{2^j}\log\frac{n}{2^j} + \frac{4}{3}\right) \\ &< kn \log n + \tfrac{4}{3}(2^k - 1) \\ &= O(n \log^2 n). \end{aligned}$$

This completes the proof.

The expected error of Algorithm 2

We continue to assume that X is a fixed $a \times b$ rectangle and the parameter t is fixed. Assuming that problem instances are drawn from $\Pi_n(X)$, we analyze the expected value of $|W_2| - |T^*|$ as a function of n. Here W_2 denotes the spanning walk generated by Algorithm 2, and T^* denotes a fixed tour.

To avoid purely technical complications we assume that $a \leqslant b < 2a$, $n = 2^{k(n)} \cdot t$, and $k(n)$ is even.

THEOREM 9. $E(|W_2|) = \sqrt{n}\,(\beta_X(t) + O(t^{-7/6}))$.

PROOF. In estimating $|W_2|$ we note that, because $a \leqslant b < 2a$, the k-stage subdivision process alternates between stages of vertical cutting and stages of horizontal cutting. Because k is even, the 2^k resulting rectangles are similar to X, but with both their dimensions scaled down by the factor $2^{-k/2}$.

The length $|W_2|$ is the sum of two contributions: (i) the sum of the lengths of the shortest tours within the rectangles; and (ii) twice the sum of the lengths of the arcs $e(Y)$.

The first contribution may be estimated as follows. The distribution of cities in any one of the 2^k rectangles is the same as if the cities had been drawn from $\Pi_t(X)$, and then all directions had been scaled down by the factor $2^{-k/2}$. Thus the expected value of the first contribution is $2^k(\beta_X(t)\sqrt{t}\,)2^{-k/2} = \beta_X(t)\sqrt{n}$.

We estimate the second contribution as follows. By Lemma 5, the expected length of $e(Y)$ is $n^{-2/3}l^{-1/3} + o(n^{-2/3}l^{-1/3})$, where l is the length of the shorter side of Y (the fact that $l'(Y)$ or $r'(Y)$ may fail to contain a city, in which case we take $|e(Y)| = 0$, only helps us). Summing over all the rectangles Y that get subdivided we have

$$\sum_{j=0}^{k/2-1} n^{-2/3}(a\cdot 2^{-j})^{-1/3}\cdot 2^{2j} + \sum_{j=0}^{k/2-1} (b\cdot 2^{-(j+1)})^{-1/3}\cdot 2^{2j+1}$$

$$= n^{-2/3}a^{-1/3}\sum_{j=0}^{k/2-1} 2^{7j/3} + 2n^{-2/3}b^{-1/3}\sum_{j=0}^{k/2-1} 2^{7(j+1)/3}$$

$$= O(n^{-2/3}2^{7k/6}) = O\left(n^{-2/3}(n/t)^{7/6}\right) = O(n^{1/2}t^{-7/6}).$$

It only remains to give the Lemma used in the proof of Theorem 9. This requires a preliminary Lemma.

LEMMA 4. *Let h points be placed at random on a unit interval. Then the expected value of the minimum distance between a pair of these points is $\leqslant 2/(h-2)(h-1)$.*

PROOF. Let A be a constant. We derive an upper bound on Q_A, the probability that the minimum distance is $\geqslant A$. Regard the points as being placed successively at random locations on the unit interval. Assuming that no two of the first k points are within A of each other, the probability that the $(k+1)$th point is within A of some previously placed point is $\geqslant 2A/2 + (k-2)A = (k-1)A$. Hence

$$Q_A \leqslant \prod_{k=2}^{h-1}(1-(k-1)A) \leqslant \prod_{k=2}^{h-1} e^{-A(k-1)} = e^{-A(h-2)(h-1)/2}.$$

The expected value of the shortest distance is

$$\int_{A=0}^{\infty} Q_A\,dA \leqslant \int_{A=0}^{\infty} e^{-A(h-2)(h-1)/2}dA = \frac{2}{(h-2)(h-1)}.$$

LEMMA 5. *Suppose cities are distributed according to a 2-dimensional Poisson distribution with density n in the infinite strip between the two parallel lines $y = 0$ and $y = l$. The expected value of the minimum distance between a city in the right half-plane and a city in the left-half plane is $\leqslant 2n^{-2/3}l^{-1/3} + o(n^{-2/3}l^{-1/3})$.*

PROOF. Let h be a positive integer to be specified later. Order the cities in increasing order of their distance from the line $x = 0$; call this total ordering '$\prec$'. Select an increasing sequence of cities $a_1, a_2, \ldots, a_h$ as follows. For a_1, we select the

first city in the ordering. Given $a_1, \ldots, a_u$, $a_{u+1} = \min\{a \mid a_u \prec a \text{ and } P_u(a) \text{ holds}\}$, where $P_u(a)$ is the property which holds if the point in $\{a_1, a_2, \ldots, a_u\}$ at the least vertical displacement from a is in the opposite half-plane from a. Among the points $\{a_1, \ldots, a_h\}$, let a_{i_1} and a_{i_2} be the two at minimum vertical distance from each other. By the way the points were selected, a_{i_1} and a_{i_2} are in opposite half-planes. We compute the expected value of the distance between a_{i_1} and a_{i_2}.

Let a_i have the coordinates (x_i, y_i). Then $|x_1|$ has an exponential distribution with mean $1/2nl$, and $|x_{i+1}| - |x_i|$ is exponential with mean $1/nl$. Thus $E(|x_i|) = (i - \frac{1}{2})/nl$. Since a_{i_1} is equally likely to be any of the a_i, $E(|x_{i_1}|) = h^{-1}\sum_{i=1}^{h}(i - \frac{1}{2})/nl = h/2nl$. Similarly, $E(|x_{i_2}|) = h/2nl$.

The random variables $\{a_1, \ldots, a_h\}$ are highly dependent, but their vertical coordinates $\{y_u\}$ are independent and uniformly distributed over $[0, l]$. For, given $a_1, a_2, \ldots, a_u$, note that a_{u+1} is the city closest to the y-axis in a region R consisting of horizontal strips in both the right-half plane and the left-half plane, such that $R \cap \{(x, y) \mid y = y_0\}$ is either $\{(x, y_0) \mid x > a_u\}$ or $\{(x, y_0) \mid x < -a_u\}$. Hence, every y_0 is equally likely to be the y-coordinate of the next city chosen. By Lemma 4, $E(|y_{i_2} - y_{i_1}|) \leqslant 2l/(h-2)(h-1)$. Hence, the expected value of the distance between a_{i_1} and $a_{i_2} \leqslant 2l/(h-2)(h-1) + h/nl$. Setting $h = [n^{1/3}l^{2/3}]$, the result follows.

It is natural to conjecture that $\beta_X(t) \geqslant \beta$ for all t. We cannot prove this, but Theorem 9 does yield the following corollary.

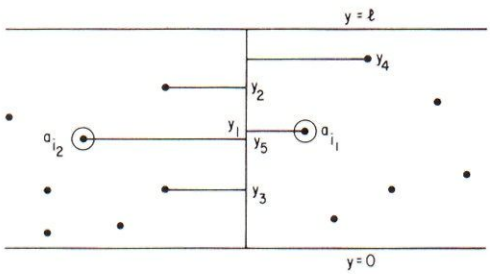

FIGURE 9. Construction in the Proof of Lemma 5 ($h = 5$).

COROLLARY 3. $\beta_X(t) + O(t^{-7/6}) \geqslant \beta$.

PROOF. Theorem 9 shows that, for infinitely many n, $\beta_X(t) + O(t^{-7/6}) \geqslant \beta_X(n)$. Since $\beta_X(n) \to \beta$, the result follows.

COROLLARY 4. *The expected value of* $|W_2| - |T^*| \leqslant \sqrt{n}\,(\beta_X(t) - \beta + O(t^{-7/6}))$.

PROOF. By Theorem 9, $E(|W_2|) = \sqrt{n}\,(\beta_X(t) + O(t^{-7/6}))$. By definition, $E(|T^*|) = \sqrt{n}\,\beta_X(n)$. By Corollary 3, $\beta_X(n) + O(n^{-7/6}) \geqslant \beta$. Combining these results, $E(|W_2| - |T^*|) \leqslant \sqrt{n}\,(\beta_X(t) - \beta_X + O(t^{-7/6}) + O(n^{-7/6}))$. Since $t < n$, this simplifies to $\sqrt{n}\,(\beta_X(t) - \beta + O(t^{-7/6}))$.

COROLLARY 5. *For every* $\alpha > 1$, *the ratio* $|W_2|/|T^*|$ *is* $\leqslant 1 + \alpha((\beta_X(t) + O(t^{-7/6}))/\beta)$, *with probability* 1.

Since $\beta_X(t) - \beta = O(t^{-1/2})$, the expected relative error for Algorithm 2 is $O(t^{-1/2})$. So far as growth rate is concerned, this is no improvement over Algorithm 1. The advantage of Algorithm 2, and our reason for presenting it in detail, is that the expected error is given explicitly in terms of $\beta_X(t)$. Thus, any information gained about $\beta_X(t)$, with respect to either its growth rate or its values for specific t, will be directly applicable.

6. Experimental results. To determine experimentally the quality of solutions produced by Algorithm 1 or Algorithm 2 would have required a supply of randomly generated problems for which optimal (or nearly optimal) solutions are known, as well as considerable investment in computer programming and data preparation. We decided instead to test the effectiveness of partitioning schemes using the minimum spanning tree problem in the plane as a substitute for the traveling-salesman problem. This permitted the experiments to be done by hand.

The problem of constructing a minimum-length spanning tree through a set of n points in the plane can be solved in $O(n \log n)$ steps [17], [18]. To test our partitioning ideas, we ignored the existence of this efficient exact solution algorithm, and instead used the following partitioning scheme, which combines features of our two partitioning algorithms for the traveling-salesman problem.

Let Y be a rectangle, oriented so that its longer sides are horizontal, and containing m cities. Then Y can be subdivided into two rectangles, $l^*(Y)$ and $r^*(Y)$, by a vertical cut equidistant between the $\lfloor m/2 \rfloor$th city from the left and the $(\lceil m/2 \rceil + 1)$th city from the left. Let $e^*(Y)$ denote the shortest segment joining a city in $l^*(Y)$ with a city in $r^*(Y)$. Let TREE(Y) be a subroutine capable of finding a minimum spanning tree through all the cities in Y; this subroutine will be called only when Y contains t or fewer cities. Let $k(n)$ be the least integer such that $2^{k(n)} \geqslant n/t$.

The following is a recursive presentation of the partitioning scheme, similar to our earlier presentations of procedures A1 and A2.

PROCEDURE A3
A3(X) = APPROXTREE(X, k)
APPROXTREE(Y, j) = *if* $j = 0$
then TREE(Y)
else APPROXTREE($l^*(Y)$, $j - 1$) $\cup$
APPROXTREE($r^*(Y)$, $j - 1$) $\cup$ $e^*(Y)$.

Applying ideas quite similar to those that occur in the analysis of Algorithm 1, we can show that the difference between $|T_{\text{approx}}|$, the length of the spanning tree produced by A3(X), and $|T_{\text{opt}}|$, the length of the minimum spanning tree, is less than $(\sum_Y \text{per}(Y)) - \text{per}(X)$, where the summation is over all rectangles Y to which the procedure TREE is applied.

Five 128-city problems were generated. In each, the cities were randomly distributed in the unit square. Each problem was solved with $t = 4$, 8, 16, 32 and 64. Table 1 reports the observed percentage error (defined as $100(|T_{\text{approx}}|/|T_{\text{opt}}| - 1)$) for each run of Algorithm 3.

TABLE 1
Percentage Error Experienced by Algorithm 3

		Problem				
		1	2	3	4	5
t	64	0.6	3.2	3.0	0.5	1.6
	32	2.4	4.7	4.3	1.7	4.9
	16	5.3	10.7	7.0	3.7	9.7
	8	7.9	14.1	8.0	6.2	12.9
	4	10.5	16.3	10.6	10.4	16.8

A good empirical formula for the error is $|T_{\text{approx}}| - |T_{\text{min}}| \approx 0.13((\sum_Y \text{per}(Y)) - \text{per}(X))$. Thus, the error is typically proportional to $(\sum_Y \text{per}\, Y) - \text{per}(X)$, but with a much smaller constant of proportionality than the one arising from a worst-case analysis.

We speculate that a similar relationship exists between the average and worst-case error of our traveling-salesman algorithms. In particular, we conjecture that $\beta_X(t) - \beta$ is proportional to $t^{-1/2}$, but with a much smaller constant of proportionality than the one given in the upper bound of Theorem 7.

7. Conclusion. We believe that the partitioning schemes presented here have both theoretical and practical interest. Using Algorithm 1 we have available a "probabilistic ϵ-approximation scheme" for the traveling-salesman problem in the plane. That is, for every $\epsilon > 0$, we can construct an algorithm $\mathfrak{A}_1(\epsilon)$ that runs within time $D_1\epsilon^2 d_1^{1/\epsilon^2} n + O(n \log n)$ and, with probability 1, constructs a tour of length $< (1 + \epsilon)$ times the cost of an optimum tour. This is done simply by running Algorithm 1, with t depending suitably on ϵ. A similar scheme can be constructed using Algorithm 2. The choice of t necessary to achieve a specified relative error in this case will depend on the rate at which $\beta_X(t)$ converges to β; our present estimates of this convergence rate (Theorem 7) are undoubtedly far too pessimistic. We conjecture that $\beta_X(t) - \beta$ is indeed proportional to $t^{-1/2}$, but with a much smaller constant of proportionality than is given by our upper estimate. Our experimental results with a partitioning algorithm for the minimum spanning tree problem lend support to this conjecture.

In practice Algorithm 1 is probably to be preferred over Algorithm 2, since its absolute error bound is independent of any assumptions about the distribution of the cities.

Some of the technical results used in analyzing the algorithms may be of independent interest. Among these results are the characterization of optimal strategies for the cutting game, and the lemma bounding the expected shortest distance between cities in adjacent rectangles.

Our partitioning schemes should prove to be of practical use in the solution of extremely large traveling-salesman problems in the plane. The heuristic methods of Krolak [11] or Lin and Kernighan [13] yield good approximate solutions in reasonable time to problems with two or three hundred cities, but they become unwieldy for larger problems. When the number of cities is in the thousands one can use a hybrid scheme which combines partitioning with one of these heuristics. In such a scheme the exact solution procedure TOUR(Y) in Algorithm 2 would be replaced by a heuristic method. It would then be feasible to run the algorithm with $t = 200$ (say), and the expected relative error would be $\Delta(t) + \beta_X(t)/\beta + O(t^{-7/6})$ where $\Delta(t)$ is the expected relative error for the underlying heuristic algorithm.

All the results generalize easily if we allow the locations of the cities to be determined according to an arbitrary 2-dimensional probability density function or if we let the domain X be any connected Lebesgue measurable set. Also, we may use the rectilinear or L_∞ metric instead of the Euclidean metric. The algorithms may also be generalized to d dimensions; the worst-case error in Algorithm 1 becomes $O((n/t)^{(d-1)/d})$, and the relative error (with probability 1) is $O(t^{-(d-1)/d})$.

Finally, the partitioning methods and their analyses may be modified in a straightforward way to apply to many other optimization problems of a geometric nature. Among these are the Steiner tree problem in the plane, the problem of constructing a minimum-weight perfect matching when the weights are Euclidean distances, the simple plant location problem in the plane, and various multi-vehicle delivery problems in the plane.

References

[1] Barbosa, L. Personal communication.

[2] Beardwood, J., Halton, J. H. and Hammersley, J. M. (1959). The Shortest Path Through Many Points. *Proc. Cambridge Philos. Soc.* **55** 299–327.

[3] Bellman, R. E. (1962). Dynamic Programming Treatment of the Travelling Salesman Problem. *J. Assoc. Comput. Mach.* **9** 61–63.

[4] Christofides, N. (1976). Worst-Case Analysis of a New Heuristic for the Travelling Salesman Problem. Abstract in *Algorithms and Complexity* (J. Traub, ed.). Academic Press.

[5] Dantzig, G. Personal communication.

[6] Garey, M. R., Graham, R. L. and Johnson, D. S. (1976). Some NP-Complete Geometric Problems. *Proc. 8th ACM Symp. on Theory of Computing* 10–29.

[7] Helbig Hansen, K. and Krarup, J. (1974). Improvements of the Held-Karp Algorithm for the Symmetric Traveling-Salesman Problem. *Math. Programming* **7** 87–96.

[8] Held, M. and Karp, R. M. (1962). A Dynamic Programming Approach to Sequencing Problems. *SIAM J.* **10** 196–210.

[9] ——— and ———. (1970). The Traveling-Salesman Problem and Minimum Spanning Trees. *Operations Res.* **18** 1138–1162.

[10] ——— and ———. (1971). The Traveling-Salesman Problem and Minimum Spanning Trees: Part II. *Math Programming* **1** 6–25.

[11] Krolak, P. D., Felts, W. and Marble, G. (1970). Efficient Heuristics for Solving Large Traveling-Salesman Problems. Presented at 7th Int. Symp. on Mathematical Programming.

[12] ———, ——— and ———. (1971). A Man-Machine Approach Toward Solving the Traveling Salesman Problem. *Comm. ACM* **14** 327–334.

[13] Lin, S. and Kernighan, B. W. (1973). An Effective Heuristic Algorithm for the Traveling-Salesman Problem. *Operations Res.* **21** 498–516.

[14] Michie, D., Fleming, J. G. and Oldfield, J. V. (1968). A Comparison of Heuristic, Interactive, and Unaided Methods of Solving a Shortest-Route Problem. *Machine Intelligence* 3. Edinburgh University Press, 245–255.

[15] Papadimitriou, C. H. (to appear). The Euclidean Traveling Salesman Problem is NP-Complete. *Theoretical Computer Science*.

[16] Rosenkrantz, D. J., Stearns, R. E. and Lewis, P. M. (1974). Approximate Algorithms for the Traveling Salesperson Problem. *Proc. 15th Annual IEEE Symp. on Switching and Automata Theory*. 33–42.

[17] Shamos, M. I. (1975). Geometric Complexity. *Proc. 7th ACM Symp. on Theory of Computing*. 224–233.

[18] ——— and Hoey, D. (1975). Closest-Point Problems. *Proc. 16th Annual IEEE Symp. on Foundations of Computer Science*. 151–162.

[19] Thompson, G. L. (1975). Algorithmic and Computational Methods for Solving Symmetric and Asymmetric Traveling Salesman Problems. Working Paper presented at the Workshop on Integer Programming, Bonn.

[20] Moran, P. A. P. (1968). *Introduction to Probability Theory*. Clarendon Press, Oxford.

COLLEGE OF ENGINEERING, UNIVERSITY OF CALIFORNIA, BERKELEY, CA 94720