SIAM J. COMPUT.
Vol. 12, No. 1, February 1983

0097-5397/83/1201-0009 $01.25/0

THE TRAVELING SALESMAN PROBLEM AND MINIMUM MATCHING IN THE UNIT SQUARE*

KENNETH J. SUPOWIT,† EDWARD M. REINGOLD‡ AND DAVID A. PLAISTED‡

Abstract. We show that the cost (length) of the shortest traveling salesman tour through n points in the unit square is, in the worst case, $\alpha_{\text{opt}}^{\text{tsp}}\sqrt{n}+o(\sqrt{n})$, where $1.075 \leqq \alpha_{\text{opt}}^{\text{tsp}} \leqq 1.414$. The cost of the minimum matching of n points in the unit square is shown to be, in the worst case, $\alpha_{\text{opt}}^{\text{mat}}\sqrt{n}+o(\sqrt{n})$, where $0.537 \leqq \alpha_{\text{opt}}^{\text{mat}} \leqq 0.707$. Furthermore, for each of these two problems there is an almost linear time heuristic algorithm whose worst case cost is, neglecting lower order terms, as low as possible.

Key words. traveling salesman problem, matching, analysis of algorithms, computational geometry, graph algorithms, heuristics

1. Introduction. Let P be a set of n points in the (Euclidean) unit square. Define a *traveling salesman tour* T of P as a set of n edges such that each point of P is an endpoint of exactly two edges, and the resulting graph (P, T) is connected. If n is even, then define a *matching* M of P as a set of $n/2$ edges such that each point of P is an endpoint of exactly one edge of M. If S is a tour or a matching then let cost(S) denote the sum of the lengths of the edges of S. The (*Euclidean*) *traveling salesman* (respectively, *matching*) *problem* is to find a minimum cost tour (respectively, matching).

The Euclidean traveling salesman problem is known to be NP-hard [7], [11] while the fastest known algorithm for Euclidean matching runs in time $\Theta(n^3)$ [6], [13]. This paper concerns fast heuristic algorithms for these two problems. Applications for heuristic Euclidean matching are described in [15].

In order to evaluate a heuristic, we use the following measure: the *worst-case performance* of a traveling salesman heuristic A is a function $f_A^{\text{tsp}}: \mathbb{N} \to \mathbb{R}$ such that

$$f_A^{\text{tsp}}(n) = \sup_P \{\text{the cost of } A\text{'s tour of } P\},$$

where P ranges over all sets of n points in the unit square. By "sup" we mean the supremum, i.e., the least upper bound; by "inf" we mean the infimum, the greatest lower bound. We use the supremum in the definition of worst-case performance because it is possible (since there are infinitely many n-point sets) that there is no n-point set P for which the cost of A's tour is maximized. If B is a matching heuristic then the worst case performance of B is the function f_B^{mat} defined analogously. The first question that arises is how good the worst-case performance of any traveling salesman (respectively, matching) heuristic can be? Let $f_{\text{opt}}^{\text{tsp}}$ denote the worst-case performance of the exhaustive optimizing traveling salesman problem algorithm. Let $f_{\text{opt}}^{\text{mat}}$ denote the worst-case performance of the $\Theta(n^3)$ optimizing matching algorithm.

* Received by the editors September 1, 1980, and in revised form May 28, 1982. Preliminary versions of some of the results contained in this paper were presented at the Twelfth Annual ACM Symposium on Theory of Computing, April; 1980. This research was supported in part by the National Science Foundation under grants NSF MCS 77-22830 and NSF MCS 79-04897.

† Hewlett-Packard Laboratories, Computer Research Center, Palo Alto, California 94304. This research was conducted while this author was at the Department of Computer Science, University of Illinois at Urbana-Champaign.

‡ Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois 61801.

We will show that both $f_{\text{opt}}^{\text{tsp}}$ and $f_{\text{opt}}^{\text{mat}}$ are $\Theta(\sqrt{n})$. Let

$$\alpha_A^{\text{tsp}} = \inf\{x : (\forall n \geqq 0)[f_A^{\text{tsp}}(n) \leqq x\sqrt{n} + o(\sqrt{n})]\},^1$$

and

$$\alpha_B^{\text{tsp}} = \inf\{x : (\forall n \geqq 0)[f_B^{\text{tsp}}(n) \leqq x\sqrt{n} + o(\sqrt{n})]\},$$

adopting the convention that the infimum of the empty set is infinity. The statement that $f_{\text{opt}}^{\text{tsp}}$ and $f_{\text{opt}}^{\text{mat}}$ are $\Theta(\sqrt{n})$ may be rephrased as $\alpha_{\text{opt}}^{\text{tsp}}$ and $\alpha_{\text{opt}}^{\text{mat}}$ are both finite and nonzero. Thus, the answer to the question of how good the worst-case performance of a heuristic can possibly be is $\alpha_{\text{opt}}^{\text{tsp}}\sqrt{n} + o(\sqrt{n})$ for the traveling salesman problem and $\alpha_{\text{opt}}^{\text{mat}}\sqrt{n} + o(\sqrt{n})$ for the matching problem.

There are two main results of this paper:

1. $1.075 \approx 2/\sqrt{\sqrt{12}} \leqq \alpha_{\text{opt}}^{\text{tsp}} \leqq \sqrt{2} \approx 1.414$, $0.537 \approx 1/\sqrt{\sqrt{12}} \leqq \alpha_{\text{opt}}^{\text{mat}} \leqq 1/\sqrt{2} \approx 0.707$.

2. There exists a heuristic algorithm A for the traveling salesman problem such that A runs in time $O(n \log n)$ and $\alpha_A^{\text{tsp}} = \alpha_{\text{opt}}^{\text{tsp}}$. Analogously, for matching there exists a heuristic algorithm B that runs in $O(n \log n)$ time and has $\alpha_B^{\text{mat}} = \alpha_B^{\text{tsp}}$.

Furthermore, if the floor function is available at unit cost, then for each unbounded, nonnegative, nondecreasing, integer-valued function f such that $f(n)$ is computable in time $O(nf(n))$, the expression "$O(n \log n)$" can be replaced by "$O(nf(n))$" in the statement of (2). Examples of such functions f are $\lceil \lg \lg n \rceil$, $\lg^* n$, $\alpha(n, n)$ [18], and so on. In other words, (2) says that for each of these two problems, there exists an almost linear time heuristic algorithm whose worst-case performance is asymptotically optimal.

The worst-case performance (as defined above) of various traveling salesman problem and matching algorithms is given in Tables 1 and 2, respectively. For matching, the rectangle algorithm is the best of the simple divide-and-conquer algorithms; its worst-case behavior is analyzed in [17] (this issue, pp. 118–143) and its average-case behavior is analyzed in [14]. The greedy algorithm for matching works by iteratively matching the two closest unmatched points; the analysis of its worst-case performance is in [1] and its $O(n^{1.5} \log n)$ implementation is in [4]. The spiral rack matching algorithm and its analysis are in [9].

Our results on worst-case performance should also be compared with the known results on *expected* performance:

(i) The expected cost of the shortest tour of n points drawn from a uniform distribution in the unit square is $\beta_{\text{tsp}}\sqrt{n} + o(\sqrt{n})$, for some β_{tsp} satisfying $0.61 \leqq \beta_{\text{tsp}} \leqq 0.92$ [2].

(ii) The expected cost of the minimum matching of n points drawn from a uniform distribution in the unit square is $\beta_{\text{mat}}\sqrt{n} + o(\sqrt{n})$, for some β_{mat} satisfying $0.25 \leqq \beta_{\text{mat}} \leqq 0.402$ [12].

2. Lower bounds on $\alpha_{\text{opt}}^{\text{tsp}}$ and $\alpha_{\text{opt}}^{\text{mat}}$. We will show that $2/\sqrt{\sqrt{12}} \leqq \alpha_{\text{opt}}^{\text{tsp}}$ and that $1/\sqrt{\sqrt{12}} \leqq \alpha_{\text{opt}}^{\text{mat}}$. Our strategy is to construct an infinite class of sets of points P such that any tour of P has cost at least $(2/\sqrt{\sqrt{12}})\sqrt{|P|}$ and any matching of P has cost at least $(1/\sqrt{\sqrt{12}})\sqrt{|P|}$. Let $k \geqq 2$ be an even integer. Let P be the set of points

$$\bigcup_{\substack{0 \leqq i \leqq k-1 \\ 0 \leqq j \leqq \lfloor 2/(\delta\sqrt{3}) \rfloor}} \left\{\left(\left[i + \frac{j \bmod 2}{2}\right]\delta, j\frac{\sqrt{3}}{2}\delta\right)\right\},$$

[1] When we say that $(\forall n)[f(n) \leqq x\sqrt{n} + o(\sqrt{n})]$, we mean that

$$(\exists g : \mathbb{N} \to \mathbb{R})[g(n) = o(\sqrt{n}) \text{ and } (\forall n)[f(n) \leqq x\sqrt{n} + g(n)]].$$

TABLE 1

Summary of results for the traveling salesman problem in the unit square, neglecting lower order terms. f is any unbounded, nonnegative, nondecreasing, integer-valued function computable in $O(nf(n))$ time. The order given for the running time assumes that the floor function is available at unit cost. If it were not, then the time for the decomposition algorithm would be $\Theta(n \log n)$.

Algorithm	Order of running time	Worst known example cost	Upper bound on worst-case performance
Optimizing	$n2^n$	$1.075\sqrt{n}$	$\alpha_{\text{opt}}^{\text{tsp}}\sqrt{n}$
Strip	$n \log n$	$1.414\sqrt{n}$	$1.414\sqrt{n}$
Decomposition	$nf(n)$	$1.075\sqrt{n}$	$\alpha_{\text{opt}}^{\text{tsp}}\sqrt{n}$

TABLE 2

Summary of results for matching in the unit square, neglecting lower order terms. f is any unbounded, nonnegative, nondecreasing, integer-valued function computable in $O(nf(n))$ time. The order given for the running time assumes that the floor function is available at unit cost. If it were not, then the times for the rectangle, spiral rack, and decomposition algorithms would be $\Theta(n \log n)$.

Algorithm	Order of running time	Worst known example cost	Upper bound on worst-case performance
Optimizing [6], [13]	n^3	$0.537\sqrt{n}$	$\alpha_{\text{opt}}^{\text{mat}}\sqrt{n}$
Greedy [1], [4]	$n^{1.5} \log n$	$0.806\sqrt{n}$	$1.075\sqrt{n}$
Strip	$n \log n$	$0.707\sqrt{n}$	$0.707\sqrt{n}$
Rectangle [17]	n	$1.436\sqrt{n}$	$1.436\sqrt{n}$
Spiral rack [9]	n	$1.014\sqrt{n}$	$1.014\sqrt{n}$
Decomposition	$nf(n)$	$0.537\sqrt{n}$	$\alpha_{\text{opt}}^{\text{mat}}\sqrt{n}$

where $\delta = 1/(k-1/2)$ is a factor introduced so that the points of P all lie in the unit square. An example is shown in Fig. 1 with $k = 6$. The points of P are vertices of a hexagonal grid, which, incidentally, also gives the densest packing of the plane by unit circles [16] and the worst known example for the greedy matching heuristic [1].

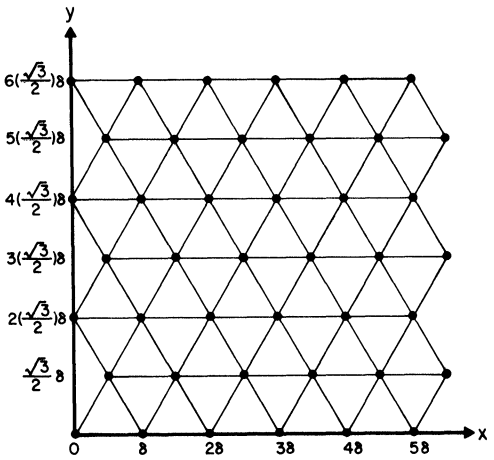

FIG. 1. *The vertices of a hexagonal grid.*

Let $n = |P|$ and let T be any tour of P. Since δ is the distance between the closest pair of points in P, each edge of T has length at least δ, so that

$$\text{cost}(T) \geqq n\delta.$$

Now

$$n = (\text{number of rows of } P) \times (\text{number of points per row})$$
$$= \left(\left\lfloor \frac{2}{\delta\sqrt{3}} \right\rfloor + 1\right) \times k > \frac{2k}{\delta\sqrt{3}}.$$

Therefore

$$\sqrt{n} > \frac{\sqrt{2k}}{\sqrt{\delta}\sqrt{\sqrt{3}}},$$

giving

$$\text{cost}(T) \geqq n\delta > \frac{\sqrt{2k}}{\sqrt{\delta}\sqrt{3}} \sqrt{n\delta} = \frac{2}{\sqrt{\sqrt{12}}} \sqrt{\delta k}\sqrt{n} > \frac{2}{\sqrt{\sqrt{12}}} \sqrt{n}.$$

Similarly, if M is a matching of P then M has $n/2$ edges, each of length at least δ, so that

$$\text{cost}(M) \geqq \frac{n}{2}\delta > \frac{1}{\sqrt{\sqrt{12}}} \sqrt{n}.$$

3. Upper bounds on $\alpha_{\text{opt}}^{\text{tsp}}$ and $\alpha_{\text{opt}}^{\text{mat}}$. We present a heuristic for the traveling salesman problem that we call the *strip algorithm*, and show that its worst-case performance is at most $\sqrt{2n} + O(1)$. The algorithm can be used for matching, when n is even, by taking the shorter of the two matchings contained in the tour found. Therefore the worst-case performance of the strip algorithm for matching is bounded above by $\sqrt{n/2} + O(1)$. This will show that $\alpha_{\text{opt}}^{\text{tsp}} \leqq \sqrt{2}$ and that $\alpha_{\text{opt}}^{\text{mat}} \leqq 1/\sqrt{2}$.

The strip algorithm for the traveling salesman problem is a modification of one analyzed for its expected performance in [2]. We are given a set of n points in the unit square. Let $r = \lceil \sqrt{n/2} \rceil$. Divide the unit square into r vertical strips, each of width $1/r$. Construct a tour T_1 of the points by starting at the lowest point in the leftmost strip, going up that strip from point to point, over to the top point of the next strip, then down that strip point by point, up the next, and so on, finally returning to the starting point, as shown by the jagged line in Fig. 2. For simplicity, not all of the input points are pictured; in order to actually have 5 strips there would have to be between 50 and 71 points.

A second tour T_2 is constructed in the same way, except that now the strip boundaries are shifted by $1/(2r)$ to the right. There are $r+1$ strips used in constructing T_2, each of width $1/r$. In Fig. 3, the strip boundaries for T_1 are shown as solid lines, those for T_2 as dashed lines. Note that the leftmost of these strips contains none of the points in its left half. Similarly, the rightmost strip contains none of the points in its right half.

The strip algorithm outputs the shorter of the two tours T_1 and T_2. The algorithm can be implemented in time $O(n \log n)$ by appropriately sorting the points.

To derive an upper bound on the cost of the tour produced, we will bound the sum of the horizontal and vertical components, and then use the triangle inequality. Consider paths P_1 and P_2 defined as follows: P_1 starts at the bottom, on the median of the leftmost of the strips used in constructing T_1. P_1 follows the median of that

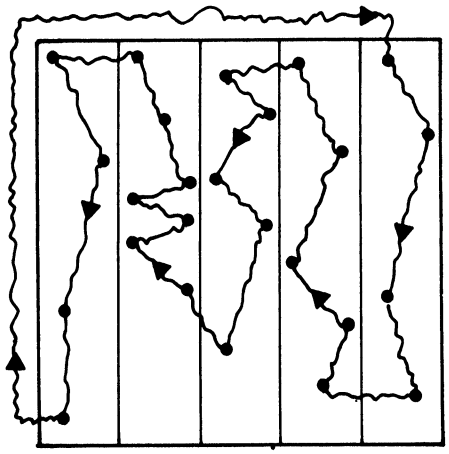

FIG. 2. *The construction of a tour, using strips.*

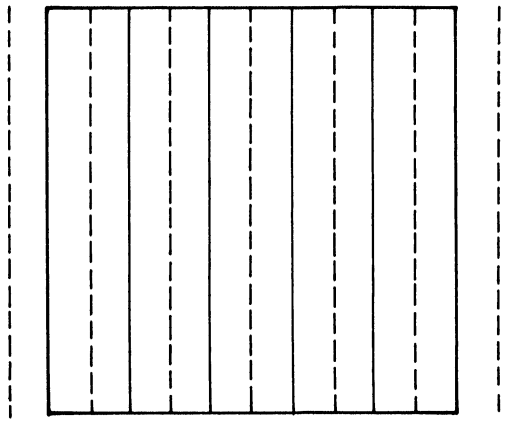

FIG. 3. *The two sets of strip boundaries.*

strip up to the top, then down the median of the next strip, up the median of the next, and so on. For each strip, for each point in that strip, the path P_1 juts out to that point and then back to the median, moving at right angles, as illustrated in Fig. 4 by the jagged line. The path P_2 is defined like P_1, except that P_2 follows the medians of the strips used to construct T_2. It follows from the triangle inequality that length $(T_1) \leqq$ length (P_1), and that length $(T_2) \leqq$ length (P_2). We now derive an upper bound on length (P_1) + length (P_2).

Consider some input point q; q must lie in some strip used for T_1 and for P_1 (shown in Fig. 5 between solid lines), and in some strip used for T_2 and for P_2 (shown in Fig. 5 between dashed lines). In Fig. 5, a segment of P_1 is shown as a bold line, and a segment of P_2 as a jagged line. It is clear that the total amount of horizontal line in P_1 or P_2 jutting out to q and back is $2 \times 1/(2r) = 1/r$. Since q was arbitrary, there is a total of n/r units of horizontal line in P_1 and P_2 together that juts out to points and back. Also, P_1 has r units of vertical line (that is, r strips of unit length). P_2 has $r+1$ strips and hence $r+1$ units of vertical line. P_1 has $1-(1/r)$ units of horizontal line that run from the end of one strip to the start of the next and P_2 has

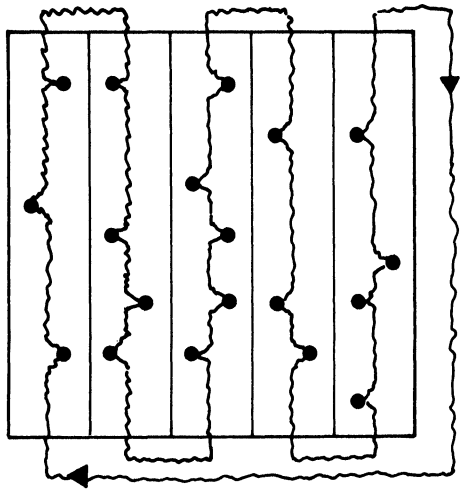

FIG. 4. *The construction of the path P_1, using strips.*

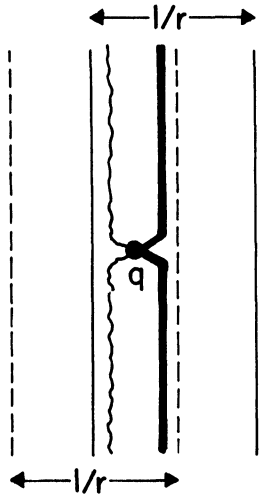

FIG. 5. *The paths P_1 and P_2 at a point q.*

1 unit of such line. Finally, P_1 and P_2 each have a segment of length at most $\sqrt{2}$ that joins the end of the last strip back to the starting position. Thus

$$\begin{aligned}\text{length}\,(T_1)+\text{length}\,(T_2) &\leqq \text{length}\,(P_1)+\text{length}\,(P_2)\\ &\leqq \frac{n}{r}+r+(r+1)+\left(1-\frac{1}{r}\right)+1+\sqrt{2}+\sqrt{2}\\ &= \frac{n}{r}+2r+O(1)\\ &= \frac{n}{\lceil\sqrt{n/2}\rceil}+2\lceil\sqrt{n/2}\rceil+O(1)\\ &= 2\sqrt{2n}+O(1).\end{aligned}$$

Therefore

$$\min\{\text{length}(T_1), \text{length}(T_2)\} \leqq \sqrt{2n} + O(1),$$

and if n is even, the cheaper of the two matchings contained in the shorter of $\{T_1, T_2\}$ has cost at most $\sqrt{n/2} + O(1)$.

These bounds are asymptotically achievable, as is suggested by the example pictured in Fig. 6. T_1 is shown as a jagged line; T_2 is not shown, but looks like T_1 shifted by $1/(2r)$ to the right. The points, which number $n = 2k^2$ for some even integer k, are arranged so that halfway between each solid vertical line and either of its two neighboring dashed vertical lines there is a vertical string of n/r points; these points are ε/r apart, for some $\varepsilon < 1/(2r)$. Intuitively, these points are placed so that T_1 and T_2 must zigzag, and hence look very much like P_1 and P_2, respectively. This attains the maximum amount (neglecting $O(1)$ terms) of horizontal line for T_1 and for T_2. There is a point at the bottom of each strip, so as to attain the maximum vertical length. To compute $\min\{\text{length}(T_1), \text{length}(T_2)\}$, note that by the Pythagorean theorem, each short, almost horizontal edge of the tour has length

$$\sqrt{\left(\frac{\varepsilon}{2r}\right)^2 + \left(\frac{1}{2r}\right)^2} > \frac{1}{2r}.$$

There are $r((n/r)-1)$ of these edges. There are r long vertical pieces, each of length $1-\varepsilon$. Recalling that $r = \lceil\sqrt{n/2}\rceil$, we have

$$\min\{\text{length}(T_1), \text{length}(T_2)\} > r\left(\frac{n}{r} - 1\right)\frac{1}{2r} + r(1-\varepsilon)$$

$$= \frac{n}{2r} - \frac{1}{2} + r - r\varepsilon > \frac{n}{2r} + r - 1 = \sqrt{2n} - 1.$$

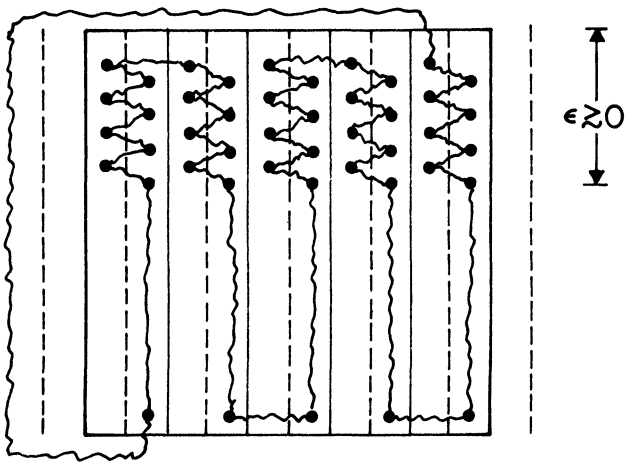

FIG. 6. *A set of points in the unit square for which length* $(T_1) \approx$ *length* $(T_2) \approx \sqrt{2n}$.

We can easily arrange the points in this example so that each of the matchings from T_1 or T_2 contains about half of the long vertical edges; hence the strip matching algorithm produces a matching for this example of cost at least $\sqrt{n/2} - \frac{1}{2}$.

4. The decomposition heuristic for the traveling salesman problem. In this section we present a decomposition heuristic for the traveling salesman problem that achieves (asymptotically) the best possible worst-case performance. The heuristic, given in Algorithm 1, is reminiscent of the traveling salesman problem heuristic given by Karp in [10]. Assuming a uniform distribution, Karp's heuristic runs in time $O(n \log n)$ almost everywhere, and, for all $\varepsilon > 0$, outputs a tour of cost at most $1+\varepsilon$ times that of the optimal tour, almost everywhere. Karp's heuristic requires exponential time on some input sets. Our heuristic, on the other hand, always runs in time $O(n \log n)$ and has the best possible worst-case performance, neglecting lower order terms. An argument similar to the one we give below proves that Karp's heuristic also has, asymptotically, the best possible worst-case performance.

In order to avoid the sorting required by the strip heuristic, Algorithm 1 uses a slightly crude approximation, the *modified strip heuristic*. It is essentially the serpentine algorithm of [9]. Each column of subsquares in the grid is a strip and we traverse the subsquares by going up the first strip, down the second, up the third, and so on. The tour thus constructed visits the points in some arbitrary order that is consistent with the cell order. Figure 7 shows an example of such a tour. The advantage of this heuristic is that it requires only $O(m)$ time for m points. It produces a tour of length $O(\sqrt{m})$ because an edge wholly contained in one of the subsquares has length at most $\sqrt{2}/c = O(1/\sqrt{m})$ (see Algorithm 1 for the definition of c).

ALGORITHM 1. The asymptotically optimal decomposition heuristic for the traveling salesman problem on a set P of n points in the unit square.

1. $c \leftarrow \lceil 2\sqrt{n}/\sqrt{\log_z f(n)} \rceil$, where $z > 2$ is some real, and $f(n)$ is a nonnegative, unbounded, nondecreasing, integer-valued function computable in $O(nf(n))$ time.
2. Divide the unit square into a regular grid of c^2 subsquares, each of side length $1/c$.
3. For each of the subsquares, do the following:
   $P' \leftarrow$ the subset of P inside the subsquare
   while $|P'| > 0$ **do**
   begin
   $k \leftarrow \min\{4\lceil n/c^2 \rceil, |P'|\}$
   $Q \leftarrow$ a set of k points chosen arbitrarily from P'
   Use dynamic programming to find the shortest traveling salesman tour of Q [3], [8].
   Distinguish one point of Q
   $P \leftarrow P' - Q$
   end
4. Perform the modified strip heuristic to find a tour of the distinguished points.
5. $T' \leftarrow$ the union of all tours found in Steps 3 and 4.
6. Convert T' to a tour T by the method of [5] (see [13])[2] and output T.

We first analyze the worst-case performance of Algorithm 1. Let α be a real number such that

$$(\forall n)[f_{\text{opt}}^{\text{tsp}}(n) \leqq \alpha\sqrt{n} + o(\sqrt{n})].$$

[2] Since T' is a union of tours, the degree of each vertex in T' is even so T' contains an Eulerian circuit. Start at an arbitrary vertex and follow the order of the Eulerian circuit, but skip any previously encountered point; the result is a Hamiltonian circuit. By the triangle inequality, the cost of this Hamiltonian circuit is no more than the cost of the Eulerian circuit we started with. The cost of the Eulerian circuit is the sum of the lengths of the edges in T'.

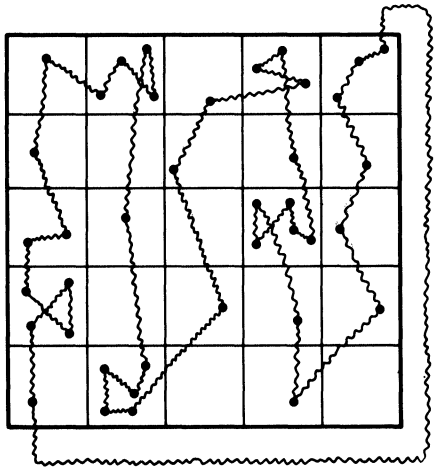

FIG. 7. *The modified strip heuristic.*

We will show that

$$(\forall n)[f^{\text{tsp}}_{\text{dec}}(n) \leqq \alpha\sqrt{n} + o(\sqrt{n})].$$

where "dec" denotes the decomposition algorithm; this will prove that $\alpha^{\text{tsp}}_{\text{dec}} = \alpha^{\text{tsp}}_{\text{opt}}$. In fact, it will prove that

$$\{x: (\forall n \geqq 0)[f^{\text{tsp}}_{\text{opt}}(n) \leqq x\sqrt{n} + o(\sqrt{n})]\} = \{x: (\forall n \geqq 0)[f^{\text{tsp}}_{\text{dec}}(n) \leqq x\sqrt{n} + o(\sqrt{n})]\},$$

which is not implied by the equality of the infima.

Fix some input set P of n points. For notational convenience, let

$$b = \lceil n/c^2 \rceil. \tag{1}$$

Number the subsquares from 1 to c^2. For all i, $1 \leqq i \leqq c^2$, let B_i denote the set of input points within the ith subsquare, and let $b_i = |B_i| \bmod 4b$. Thus the number of applications of the optimizing dynamic programming algorithm (that is, the number of executions of the body of the **while** loop) when working on subsquare B_i is at most

$$\frac{|B_i| - b_i}{4b} + 1.$$

Let

$$t = \sum_{i=1}^{c^2} \frac{|B_i| - b_i}{4b} = (n - \sum_{i=1}^{c^2} b_i)/(4b); \tag{2}$$

thus $t + c^2$ is the total number of executions of the body of the **while** loop.

Now for all $r \geqq 1$, the cost of the tour produced by the optimizing algorithm on r points in a $1/c$ by $1/c$ square is at most $[\alpha\sqrt{r} + o(\sqrt{r})]/c$. The factor $1/c$ scales down the cost from the unit square to the $(1/c) \times (1/c)$ square. Therefore the sum of the costs of all the tours produced by the optimizing algorithm is at most

$$\frac{1}{c}\left[t(\alpha\sqrt{4b} + o(\sqrt{b})) + \sum_{i=1}^{c^2} (\alpha\sqrt{b_i} + o(\sqrt{b_i}))\right] = \frac{\alpha}{c}\left[t\sqrt{4b} + \sum_{i=1}^{c^2} \sqrt{b_i} + o(c^2\sqrt{b})\right],$$

since $b_i \leqq 4b$ and $t \leqq c^2$.

The tour produced in Step 4 by the modified strip algorithm on at most c^2 points has cost $O(c)$. In Step 5, the tour T produced by the method of [5] (see [13]) from T', the union of the tours found in Steps 3 and 4, has cost at most $\sum_{e \in T'} \text{length}(e)$, by the triangle inequality. Therefore the total cost of the tour T produced by the algorithm is at most

$$\frac{\alpha}{c}\left[t\sqrt{4b} + \sum_{i=1}^{c^2} \sqrt{b_i} + o(c^2\sqrt{b})\right] + O(c). \tag{3}$$

Note that $\sqrt{b} = \sqrt{\lceil n/c^2 \rceil} < \sqrt{n}/c + 1$ and that $c = o(\sqrt{n})$ so (3) can be rewritten as

$$\frac{\alpha}{c}\left[t\sqrt{4b} + \sum_{i=1}^{c^2} \sqrt{b_i}\right] + o(\sqrt{n}). \tag{4}$$

Let $g : \mathbb{R}^{c^2} \to \mathbb{R}$ be defined by

$$g(b_1, b_2, \cdots, b_{c^2}) = t\sqrt{4b} + \sum_{i=1}^{c^2} \sqrt{b_i}$$

$$= \frac{1}{4b}\left(n - \sum_{i=1}^{c^2} b_i\right)\sqrt{4b} + \sum_{i=1}^{c^2} \sqrt{b_i} = \frac{1}{\sqrt{4b}}\left(n - \sum_{i=1}^{c^2} b_i\right) + \sum_{i=1}^{c^2} \sqrt{b_i}.$$

Taking partial derivatives shows that g is maximized at $b_1 = b_2 = \cdots = b_{c^2} = b$. In this case $n \geqq bc^2$, but because $b = \lceil n/c^2 \rceil$, we have $bc^2 \geqq n$ so that $n = bc^2$ giving

$$t = \left(n - \sum_{i=1}^{c^2} b_i\right)/(4b) = (n - bc^2)/(4b) = 0.$$

Therefore (4) is maximized when $t = 0$ and $b_1 = b_2 = \cdots = b_{c^2} = b$. Hence

$$\text{cost}(T) \leqq \frac{\alpha}{c} \sum_{i=1}^{c^2} \sqrt{b} + o(\sqrt{n}) = \alpha c\sqrt{b} + o(\sqrt{n}) = \alpha\sqrt{n} + o(\sqrt{n}),$$

since $\sqrt{b} < \sqrt{n}/c + 1$. Thus

$$f_{\text{dec}}^{\text{tsp}}(n) \leqq \alpha\sqrt{n} + o(\sqrt{n})$$

so that $\alpha_{\text{dec}}^{\text{tsp}} = \alpha_{\text{opt}}^{\text{tsp}}$ as claimed. Thus the decomposition algorithm has the asymptotically best possible worst-case performance.

We now analyze the running time of Algorithm 1. Under the real RAM model of computation, partitioning the n points into the c^2 subsquares can be done in $O(n \log n)$ time, since the subsquares form a grid. If we allow the floor function at unit cost, then this partitioning can be done in $O(n)$ time.

There is a dynamic programming algorithm that finds the shortest tour of r points in time $O(r2^r)$ [3], [8], hence in time $O(z^r)$ for $z > 2$. Step 3 makes at most $t + c^2$ calls on that algorithm, each with at most $4b$ points. Therefore the time required by Step 3 is

$$O((t + c^2)z^{4b}) = O\left(\left[\left(n - \sum_{i=1}^{c^2} b_i\right)\Big/(4b) + c^2\right]z^{4b}\right)$$

$$= O\left(\left(\frac{n}{b} + c^2\right)z^{4b}\right)$$

$$= O(c^2 z^{4\lceil n/(4n/\log_z f(n))\rceil}) = O\left(\frac{nf(n)}{\log f(n)}\right) = O(nf(n)).$$

There are at most

$$t+c^2<\frac{n}{4b}+c^2=O\left(\frac{n}{\log f(n)}\right)$$

points distinguished in Step 3. Therefore Step 4 can be performed in time $O(n/\log f(n))$.

Thus, the total running time of Algorithm 1 is $O(n \log n)$ under the real RAM model of computation. If the floor function is available at unit cost, the running time is $O(nf(n))$.

5. The decomposition heuristic for matching. In this section we present a decomposition heuristic for matching that, like Algorithm 1 for the traveling salesman problem, achieves (asymptotically) the best possible worst-case behavior. The heuristic, given in Algorithm 2, is almost an exact parallel to Algorithm 1, and its analysis is virtually identical. In particular, we can show, with the same argument as before, that if α is a real number such that

$$(\forall n)[f_{\text{opt}}^{\text{mat}}(n)\leqq\alpha\sqrt{n}+o(\sqrt{n})],$$

then

$$(\forall n)[f_{\text{dec}}^{\text{mat}}(n)\leqq\alpha\sqrt{n}+o(\sqrt{n})]$$

so that $\alpha_{\text{dec}}^{\text{mat}}=\alpha_{\text{opt}}^{\text{mat}}$.

ALGORITHM 2. The asymptotically optimal decomposition heuristic for matching.

1. $c\leftarrow\lceil\sqrt{n}/\sqrt{f(n)}\rceil$, where $f(n)$ is a nonnegative, unbounded, nondecreasing, integer-valued function computable in $O(nf(n))$ time.
2. Divide the unit square into a regular grid of c^2 subsquares, each of side length $1/c$.
3. For each of the subsquares, do the following:
   $P'\leftarrow$ the subset of P inside the subsquare
   if $|P'|$ is odd **then** distinguish an arbitrarily chosen point in P' and delete it from P'
   while $|P'|>0$ **do**
   begin
   $k\leftarrow$ the largest even integer less than or equal to $\min\{4\lceil n/c^2\rceil, |P'|\}$
   $Q\leftarrow$ a set of k points chosen arbitrarily from P'
   Use the optimizing matching algorithm [6], [13] to find the minimum cost matching of Q
   $P\leftarrow P'-Q$
   end
4. Perform the modified strip heuristic to find a tour of the distinguished points, then find the less costly of the two matchings contained in the tour.
5. Output the union of all matchings found in Steps 3 and 4.

As for the time required, the partitioning takes $O(n \log n)$ under the real RAM model of computation and $O(nf(n))$ if the floor function is available at unit cost: Let b and t be defined by (1) and (2), respectively; note that now $b=\Theta(\sqrt{f(n)})$. There are at most $t+c^2$ calls on the optimizing algorithm, each with at most $4b$ points and hence each requiring $O(b^3)$ time. Thus Step 3 requires time

$$O((t+c^2)b^3)=O\left(\left(\frac{n}{b}+c^2\right)b^3\right)=O(c^2b^3)=O(nf(n)).$$

There is at most one distinguished point in each subsquare, so Step 4 can be performed in time $O(c^2) = O(n/\sqrt{f(n)})$. The total time for Algorithm 2 is thus $O(n \log n)$ without the floor function and $O(nf(n))$ with it.

6. Open problems. Many questions remain unanswered. We know that

$$\frac{2}{\sqrt{\sqrt{12}}} \leqq 2\alpha_{\text{opt}}^{\text{mat}} \leqq \alpha_{\text{opt}}^{\text{tsp}} \leqq \sqrt{2}.$$

Does $\alpha_{\text{opt}}^{\text{tsp}} = 2\alpha_{\text{opt}}^{\text{mat}}$? We can define $\alpha_{\text{opt}}^{\text{mst}}$ for the minimum spanning tree problem in analogy to $\alpha_{\text{opt}}^{\text{tsp}}$ and $\alpha_{\text{opt}}^{\text{mat}}$; the hexagonal grid example of Fig. 1 establishes that $\alpha_{\text{opt}}^{\text{mst}} \geqq 2/\sqrt{\sqrt{12}}$. Furthermore, it is obvious that $\alpha_{\text{opt}}^{\text{mst}} \leqq \alpha_{\text{opt}}^{\text{tsp}}$. Does $\alpha_{\text{opt}}^{\text{mst}} = \alpha_{\text{opt}}^{\text{tsp}}$? How does $2\alpha_{\text{opt}}^{\text{mat}}$ compare with $\alpha_{\text{opt}}^{\text{mst}}$? We conjecture that

$$\alpha_{\text{opt}}^{\text{tsp}} = \alpha_{\text{opt}}^{\text{mst}} = 2\alpha_{\text{opt}}^{\text{mat}} = \frac{2}{\sqrt{\sqrt{12}}}.$$

Finally, our decomposition algorithms are not quite linear time; are there linear time algorithms A and B for the traveling salesman problem and matching, respectively, for which $\alpha_A^{\text{tsp}} = \alpha_{\text{opt}}^{\text{tsp}}$ and $\alpha_B^{\text{mat}} = \alpha_{\text{opt}}^{\text{mat}}$?

7. Acknowledgment. We gratefully acknowledge suggestions by the referee that helped sharpen one of the time bounds, improve the notation, and clarify the exposition.

REFERENCES

[1] D. AVIS, *Worst case bounds for the Euclidean matching problem*, Internat. J. Comput. Math. Appl., 7 (1981), pp. 251–257.

[2] J. BEARDWOOD, J. H. HALTON AND J. M. HAMMERSLEY, *The shortest path through many points*, Proc. Cambridge Phil. Soc., 55 (1959), pp. 299–327.

[3] R. BELLMAN, *Dynamic programming treatment of the travelling salesman problem*, J. Assoc. Comput. Mach., 9 (1962), pp. 61–63.

[4] J. L. BENTLEY AND J. B. SAXE, *Decomposable searching problems*, 1: *Static-to-dynamic transformation*, J. Algorithms, 1 (1980), pp. 301–358.

[5] N. CHRISTOFIDES, *Worst-case analysis of a new heuristic for the travelling salesman problem*, Technical Report of the Graduate School of Industrial Administration, Carnegie–Mellon Univ., Pittsburgh, PA, 1976.

[6] H. GABOW, *An efficient implementation of Edmond's algorithm for maximum matching on graphs*, J. Assoc. Comput. Mach., 23 (1976), pp. 221–234.

[7] M. R. GAREY, R. L. GRAHAM AND D. S. JOHNSON, *Some* NP*-complete geometric problems*, in Proc. Eighth ACM Symposium on Theory of Computing, 1976, pp. 10–22.

[8] M. HELD AND R. M. KARP, *A dynamic programming approach to sequencing problems*, J. Soc. Indust. Appl. Math., 10 (1962), pp. 196–210.

[9] M. IRI, K. MUROTA AND S. MATSUI, *Linear-time approximation algorithms for finding the minimum-weight perfect matching on a plane*, Inform. Process. Lett., 12 (1981), pp. 206–209.

[10] R. M. KARP, *The probabilistic analysis of some combinatorial search algorithms*, in Algorithms and Complexity: New Directions and Recent Results, J. F. Traub, ed., Academic Press, New York, 1977.

[11] C. H. PAPADIMITRIOU, *The Euclidean traveling salesman problem is* NP*-complete*, Theoret. Comput. Sci., 4 (1977), pp. 237–244.

[12] ———, *The probabilistic analysis of matching heuristics*, in Proc. Fifteenth Annual Allerton Conf. on Communication, Control and Computing, 1977, pp. 368–378.

[13] C. H. PAPADIMITRIOU AND K. STEIGLITZ, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1982.

[14] E. M. REINGOLD AND K. J. SUPOWIT, *Probabilistic analysis of divide and conquer heuristics for minimum weighted Euclidean matching*, Networks, to appear.

[15] E. M. REINGOLD AND R. E. TARJAN, *On a greedy heuristic for complete matching*, this Journal, 10 (1981), pp. 676–681.
[16] C. A. ROGERS, *Packing and Covering*, Cambridge Univ. Press, Cambridge, 1964.
[17] K. SUPOWIT AND E. M. REINGOLD, *Divide and conquer heuristics for minimum weighted Euclidean matching*, this Journal, this issue, pp. 118–143.
[18] R. E. TARJAN, *Efficiency of a good but not linear set union algorithm*, J. Assoc. Comput. Mach., 22 (1975), pp. 215–225.