A Survey of Heuristics for the Weighted Matching Problem

David Avis
School of Computer Science, McGill University, Montreal, Canada

This survey paper reviews results on heuristics for two weighted matching problems: matchings where the vertices are points in the plane and weights are Euclidean distances, and the assignment problem. Several heuristics are described in detail and results are given for worst-case ratio bounds, absolute bounds, and expected bounds. Applications to practical problems and some mathematical complements are also included.

I. INTRODUCTION

Edmonds' algorithm [10] for finding a minimum weighted matching in a graph has stood for almost two decades as one of the paradigms in the search for polynomial-time algorithms for integer programming problems. Given the plethora of combinatorial problems for which no such algorithm is known, it is little wonder that heuristics for solving the weighted matching problem have not received much attention during this time. However, in the past few years, several important developments have occurred in the applications area, where large-scale problems have meant that even an $O(n^3)$ implementation of Edmonds' algorithm (see, e.g., [18]) is too expensive. On the theoretical side also, the inherent complexity of Edmonds' algorithm has resisted analysis of the expected weight of the optimal solution, given a known distribution of the edge weights. Suboptimal but conceptually simpler heuristics are easier to analyze and provide bounds on this expected value.

In this survey paper, we review the rapid progress that has been made in the analysis of matching heuristics during the last few years. The remainder of this section contains an outline of the paper and a description of some of the applications that motivate the study of matching heuristics. An attempt is made in this section to present a brief historical summary of the relevant papers, and to present the applications in some detail. The remainder of the paper is divided into two main sections: weighted matchings where the weights are Euclidean distances and matchings where the weights are arbitrary positive numbers. In each of these sections, representative heuristics are described along with the most recent results on their analyses. Unfortunately not all heuristics can be included: there have been well over twenty heuristics proposed for the Euclidean case alone. It is hoped that the heuristics presented are a good representation of this group, and that the interested reader will be able to explore the

NETWORKS, Vol. 13 (1983) 475-493

CCC 0028-3045/83/040475-19$02.90

original papers to discover those heuristics that space did not allow us to include. Where available, numerical results are included that compare various heuristics on empirical data. Finally, in each section we attempt to connect the results obtained with related theoretical questions of a primarily mathematical interest. The fourth section of this survey contains conclusions on which heuristics are likely to be the most useful in practice, a summary of comparative results, and a list of some open problems.

Let us now return to outline the development of heuristics for the weighted matching problem. An early paper is that of Kurtzberg [15], who studied heuristics for the assignment problem. When the weights are uniformly distributed in the interval [0,1], he exhibits heuristics for which the expected minimum weight of the perfect matching found is at most $\ln n$. This opened the question of finding the expected value of the optimum matching. Further progress was made by Donath [9], who presented experimental evidence to show that this expected value was in fact bounded by a constant of about 1.6, independent of n, a most remarkable observation. In 1977, this question was settled when Walkup [26, 27] showed that the expected weight of the optimal matching was bounded above by 3. Walkup's proof was, however, nonconstructive. Most recently, Lai [16], in his thesis, has given a constructive proof of Walkup's result by exhibiting an $O(n^2)$ heuristic which finds a solution to the assignment problem with an expected weight of at most 6. Heuristics for weighted matchings in the complete graph K_n are discussed in Avis [3]. This paper contains average- and worst-case results for two "greedy" heuristics which repeatedly match vertices adjacent along an edge of lowest possible cost.

The above papers all concern matchings where the edge weights are arbitrary. Further, several of the papers were motivated by theoretical considerations. A separate literature has arisen around the problem of finding a minimum-weight matching in K_n where the vertices are points in the Euclidean plane and where the Euclidean distance between two points is the weight assigned to the corresponding edge. Since these weights satisfy the triangle inequality, much stronger results are possible. This series of papers arises from the problem of determining the efficient use of a mechanical plotter, an application which appears to have been discovered in North America and independently rediscovered later in Japan.

Consider the problem of plotting a graph $G = (V, E)$ where the vertices are given by their (x, y) coordinates. Figure 1 contains a typical example, a map of the major roads in Tokyo. If the graph contains an Eulerian cycle or path then it can be drawn with no wasted pen movement. Since a graph has an Eulerian cycle or path if and only if it has at most two vertices of odd degree, this condition can be easily checked. Furthermore, such a cycle or path can easily be found in linear time using depth-first search (see, e.g., [18]). Otherwise the pen must be moved in the "up" position, resulting in wasted pen motion. The graph G can be made Eulerian by matching the edges of odd degree, but remembering to lift the pen "up" when these matched edges are traversed. If the vertices are matched arbitrarily, the wasted pen movement can be quite large. Figure 2(i) shows the pen movement wasted in using this technique, by the device of reversing the pen "up" and "down" instructions. Thus the lines plotted represent the wasted pen movement. Finding a minimum-weight matching will reduce the waste to a minimum: this is illustrated in Figure 2(iii). Large plotting problems can involve

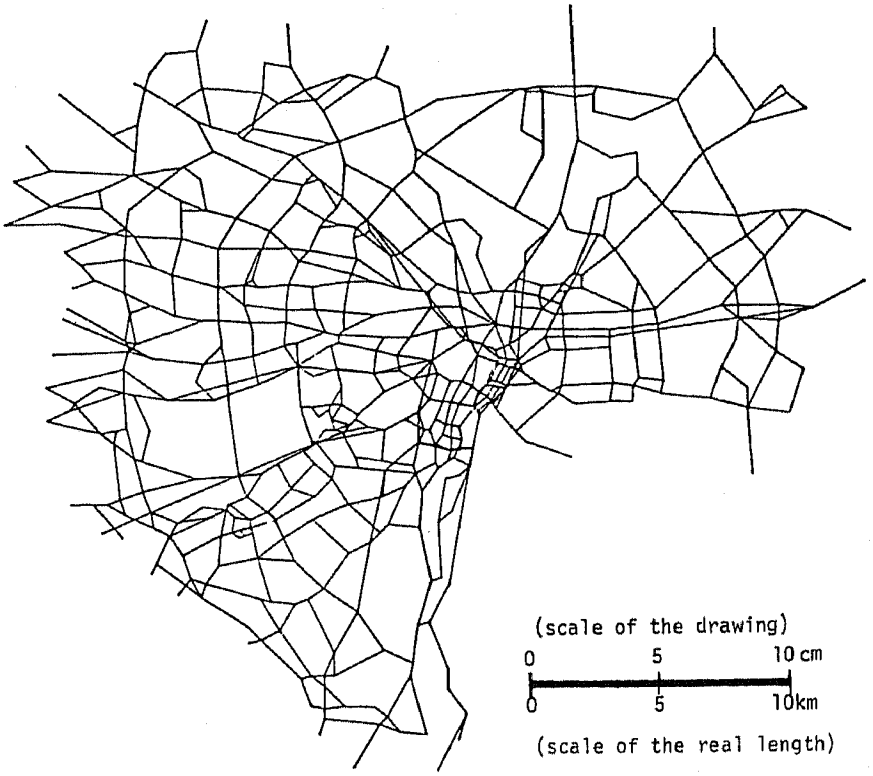

FIG. 1. A street map of Tokyo (courtesy Iri et al. [13]).

thousands of vertices (e.g., VLSI circuits), and so the computation required to compute this matching would be much more expensive than the wasted plotter time. A linear-time heuristic known as the spiral rack was implemented and results in the pen wastage shown in Figure 2(ii). This heuristic takes about 0.1% of the CPU time required for the optimal solution, and produces a matching of weight about 50% greater. This compares with the approach in Figure 2(i) where a matching of weight some five times greater than optimal was found.

The reader may have noted that the above problem is none other than the Chinese postman problem in a different guise. The plotter application appears in Reingold and Tarjan [20] and Iri and Taguchi [11]. The former reference describes the worst-case ratio of a "greedy" heuristic to the optimal solution. The latter paper describes some experiments on the example of Figure 1. In Avis [4], a case is made for an alternative worst-case measure based on the absolute difference between the heuristic and optimal solutions. An analysis is given of the greedy heuristic under this measure. Both measures are discussed in the comprehensive paper of Supowit et al. [23], which appears in more detail as [24] and [25]. In Iri et al. [12, 13] extensions are made to results in their earlier preliminary paper, describing linear heuristics. Finally, in Reingold and Supowit [19], some average-case results are given for the heuristics described in their earlier papers. Expected-case results for a different heuristic had previously appeared in Papadimitriou [17], although not in the context of the plotting problem. A new proof of this and related results is contained in Steel [22]. Little of qualitative nature is known about the optimum Euclidean matching. Such information would be useful in the design of fast heuristics. Akl [2] has examples to show that the optimum matching is not a subgraph of either the Delaunay or minimum-weight *triangulation*.

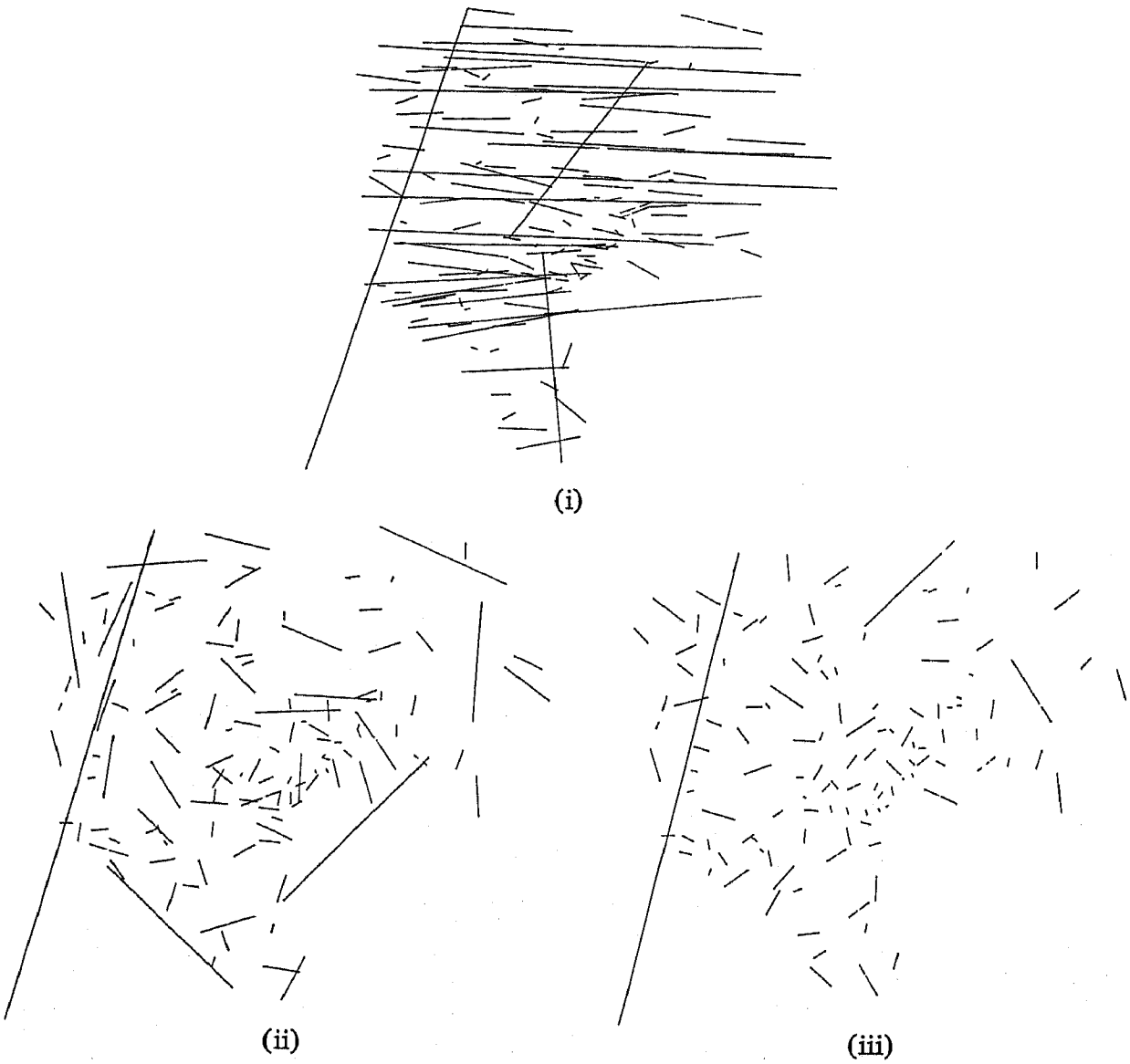

FIG. 2. Wasted pen movement for various matching strategies (courtesy Iri et al. [13]). (i) points matched in arbitrary order. (ii) points matched by SPIRAL RACK heuristic. (iii) points matched by optimal algorithm.

Matching heuristics also find application as subprocedures in heuristics for the traveling salesman problem (TSP). Christofides' heuristic for the TSP involves finding a matching in K_n ([8], [18]). Similarly, the assignment problem appears as a subproblem in a heuristic for the TSP due to Karp [15]. Since both of these procedures for the TSP are approximate, it seems worthwhile to consider using heuristics for the matching subproblem, reducing the overall running time. This allows for several heuristic solutions to be tried within the original time bound. Further discussion of these topics as well as the application of local neighborhood search to matching problems can be found in [1]. An example by Friese (private communication) shows that the worst-case ratio bound of this method is very bad [$\Omega(n^2)$].

We conclude this section by remarking that while most of the results in this survey have appeared previously, there are some new results. In particular, the analysis of Kurtzberg's decomposition heuristic in Sections III.A-III.C is new. In addition, Theorem 4 of Section III.B generalizes a result of [3].

II. HEURISTICS FOR EUCLIDEAN MATCHING

A. Description of Heuristics

Let n be an even integer and let $P = \{p_1, \ldots, p_n\}$ be a set of n points in the plane. A (perfect) *matching* M is a set of $\frac{1}{2}n$ edges connecting points in P such that each point in P is the endpoint of exactly one edge. The cost of M is the sum of lengths of edges in M. Normally we will assume the lengths are given in the Euclidean L_2 metric, although some results will also be presented for the L_∞ metric, which is often a more appropriate measure for the plotting application mentioned in Section I. We will sometimes add the additional condition that P is a subset of a bounded Euclidean region, for example, the unit square.

In this section we describe four heuristics that have received significant analysis and span the range of time complexities from "medium" speed to linear, in terms of the given number of points. The first heuristic, GREEDY, is perhaps the most obvious:

```
procedure GREEDY
  begin
    while P contains at least two points
      do begin
        Choose the closest pair (p_i, p_j) ∈ P;
        Add (p_i, p_j) to M;
        Delete p_i, p_j from P
      end
  end;
```

Although easily stated, the worst-case time complexity of GREEDY is not immediately obvious. In discussing time complexity, we shall use the usual real RAM model with floor function $[x]$ *available* at unit cost. Under this model it is straightforward to implement GREEDY in $O(n^2 \log n)$ time by first computing and sorting all edge lengths and then by scanning this sorted list once. In fact GREEDY can be implemented in $O(n^{3/2} \log n)$ time as shown in [7], and this is the best known time complexity although it seems unlikely that this is optimum. The GREEDY heuristic appears in [3, 4, 15, 20].

All of the other heuristics we consider apply to the situation where the points in P lie in a bounded region. The first such utilizes the notion of dividing the region into strips. This idea has been useful in analyzing the traveling salesman problem [6], although its application to the matching problem occurs in [17]. The variation described here, which improves the worst-case performance without increasing the time complexity, is taken from [23, 25].

```
procedure STRIP
  begin
    1. Set r = [½n] and divide the unit square into r vertical strips.
    2. Form paths P_1, ..., P_r by joining each point in strip i to the point with the
       next-greatest vertical coordinate.
    3. Link the paths to form a traveling salesman tour T_1 by joining the top element
```

of P_1 to the top of P_2, the bottom element of P_2 to the bottom element of P_3, etc. Finally link the two remaining endpoints.

4. Form a new partition of the unit square by shifting the partition of step 1 by $\frac{1}{2}r$ to the right; repeat steps 2 and 3 for this partition, obtaining a new tour T_2.
5. Each of the tours T_1 and T_2 contains two matchings; choose the smallest of the four matchings.

end;

STRIP is illustrated in Figure 3. In analyzing the time complexity of STRIP it is apparent that the partition into strips in Steps 1 and 4 can be performed in $O(n)$ time, using the floor function. Step 2 can be implemented by sorting the points in each strip by their vertical coordinate, at a cost of $O(n \log n)$. Steps 3 and 5 take $O(n)$ time, thus the overall time complexity of STRIP is $O(n \log n)$, making it considerably faster than GREEDY.

Our third heuristic, again due to Supowit et al. [24], also applies to a bounded region, which will be assumed to be of dimension $\sqrt{2} \times 1$ for convenience in describ-

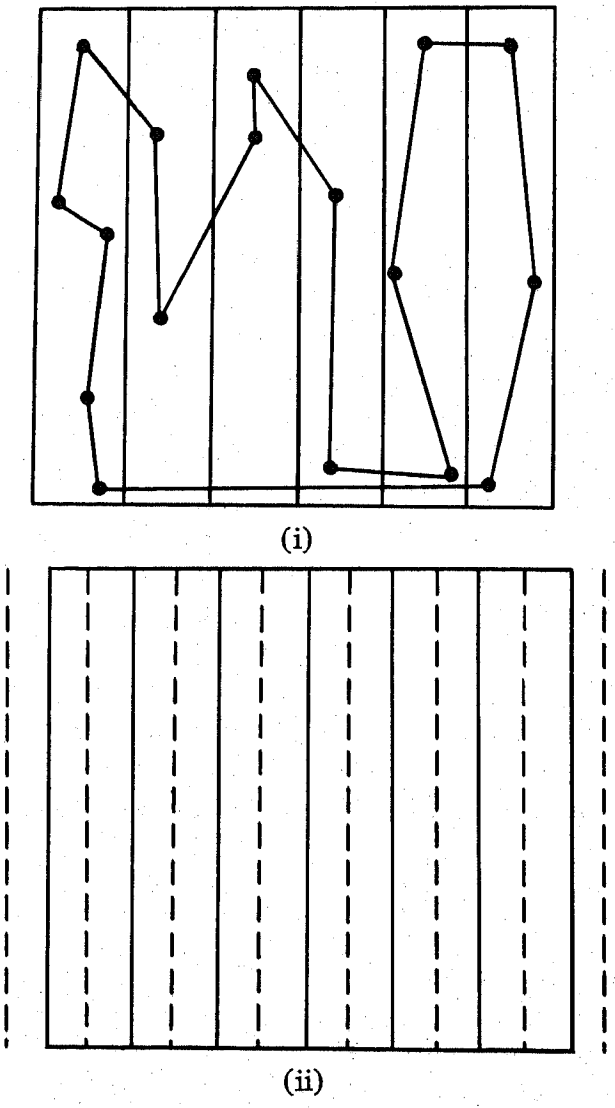

FIG. 3. (i) Tour T_1 produced by STRIP. (ii) The two sets of STRIP boundaries.

ing the algorithm. Applications to other region sizes are straightforward. This heuristic, called λ-RECTANGLE, is recursive and employs the parameter λ to control the depth of the recursion. The special case when $\lambda = [\log n]$ is particularly important, and in this case we drop the parameter and refer to the RECTANGLE heuristic. The variable "level" measures the depth of the recursion and is initially zero. R is the rectangle containing the points to be matched.

```
procedure λ-RECTANGLE (level, R)
  begin
    if level ≤ λ and R contains at least two points
      then begin
        Bisect R to form rectangles R1 and R2 having sides of ratio √2:1;
        λ-RECTANGLE (level+1, R1);
        λ-RECTANGLE (level+1, R2);
        if R1 and R2 contain unmatched points pi and pj,
          add (pi, pj) to M
          end
      else
        arbitrarily match the points in R until at most one point is left unmatched
  end;
```

The recursive version of RECTANGLE given above can be implemented in time $O(n \log n)$ without using the floor function. With the floor function, RECTANGLE can be implemented in a bottom-up fashion nonrecursively in time $O(n)$ [24]. Note that at level $[\log n]$ there are approximately n rectangles into which points must be placed. The analysis of RECTANGLE appears in [19, 23, 24].

Our final heuristic has been dubbed the "spiral rack" by its inventors, Iri et al. [12], and relies on a partition of the unit square into an array of $n^{1/2} \times n^{1/2}$ subsquares. The name derives from the order in which the subsquares are visited, which is illustrated in Figure 4. For convenience in describing the algorithm, we assume $k = \alpha n^{1/2}$ is an integer where α is a parameter of the algorithm to be determined.

```
procedure SPIRAL RACK
  begin
    Divide the points into an array of k^2 subsquares of dimension 1/k × 1/k.
    Order the cells according to the spiral rack order (Fig. 4), forming the sequence
      C1, C2, . . . , Cn.
    for i = 1 to n
      do begin
        if Ci ≠ ∅ form a path arbitrarily through the points in Ci;
        if i > 1 link this path to a vertex of degree 1 in Ci-1
        end
    Link the two remaining vertices of degree 1, forming a traveling salesman tour.
    Choose the smaller of the two matchings that form the tour.
  end;
```

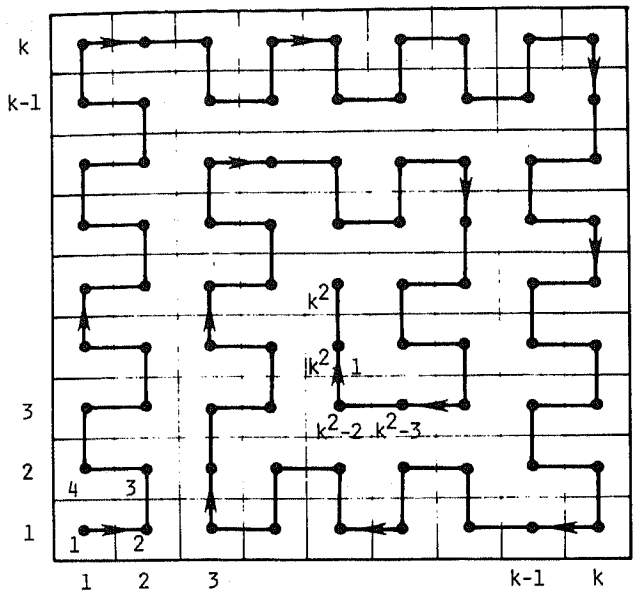

FIG. 4. SPIRAL RACK cell order (courtesy Iri et al. [13]).

The performance of SPIRAL RACK can be improved by the addition of the following preprocessing step: first match as many points as possible within each cell separately, then apply SPIRAL RACK to the remaining points. The parameter α has a substantial influence upon both the worst-case and average-case behavior of SPIRAL RACK. For practical purposes, the authors suggest a value of $\alpha = 1.29$ for the L_2 metric and $\alpha = 1.26$ for the L_∞ metric [13].

It is straightforward to implement this heuristic in $O(n)$ time using the floor function. Other linear heuristics can be constructed by altering the order in which the cells are visited. An alternative ordering is shown in Figure 5, and has been dubbed the "serpent" ordering.

This concludes the description of four heuristics that have appeared in the recent literature. The next subsections contain a synopsis of known results concerning these

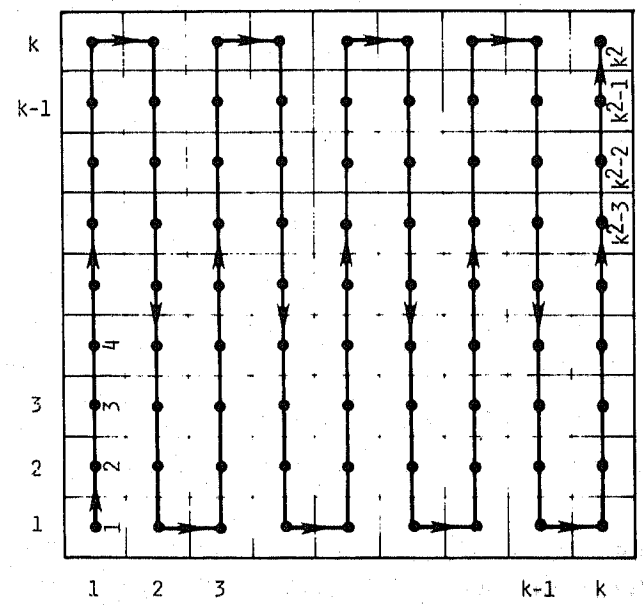

FIG. 5. SERPENT cell order (courtesy Iri et al. [13]).

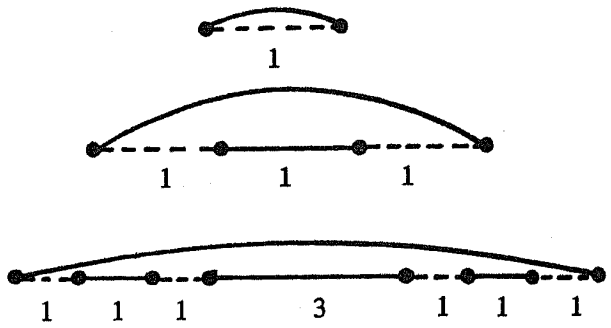

FIG. 6. First three of a series of examples to show that the bound in Theorem 1 is sharp (from [20]): solid edges, GREEDY solution; dashed edges, optimal solution.

heuristics. To this end we require some further notation. Let $C_{GR}(P)$, $C_S(P)$, $C_R(P)$, $C_{SR}(P)$, $C_{MIN}(P)$ be the cost of the matching obtained by GREEDY, STRIP, RECTANGLE, SPIRAL RACK, and the optimum algorithm, respectively. Since the dependence on P is usually clear, we will normally write C_{GR}, C_S, C_R, C_{SR}, and C_{MIN} for brevity.

B. Worst-Case Results: Ratio Bounds

One of the most common performance measures for a heuristic bounds the ratio of the worst-case costs of the heuristic and optimal solutions in terms of a function of the size of the input:

$$\frac{C_{HEUR}(P)}{C_{MIN}(P)} \leqslant f(|P|). \tag{1}$$

Ideally, $f(|P|)$ is a small constant independent of P. This type of bound is particularly useful when the size of the input space is unbounded, implying that the solution is unbounded even for fixed $|P|$.

The only heuristic from Section II.A for which a ratio bound is known is GREEDY, for which Reingold and Tarjan proved

Theorem 1 [20].

$$C_{GR}/C_{MIN} \leqslant \tfrac{4}{3} n^{\lg 1.5}.$$

Here lg represents the logarithm to the base 2. The bound is in fact sharp as illustrated by the examples in Figure 6. This result shows that the greedy solution can be very much larger than the optimum solution, even for rather small values of $n = |P|$. A constant-ratio bound of $\frac{1}{2}$ is known for the *maximum* weighted matching problem using the GREEDY heuristic modified in the obvious way. This bound holds independent of the Euclidean metric, and so is deferred to Section III.B.

C. Worst-Case Results: Absolute Bounds

In the plotting application described in Section I the cost of wasted plotter movement must be balanced against the CPU cost of finding a good matching. The cost of

increased plotter time is proportional to the difference $C_{HEUR} - C_{MIN}$ rather than to the ratio. The example of Figure 6 does not therefore result in much wasted pen movement since the points are collinear: C_{GR} is bounded by the diameter of the plotting region, which is often constant in practice. It is therefore of interest to study the behavior of the various heuristics in regions of bounded size, the unit square being the most convenient such region. All four heuristics have been analyzed to find upper bounds on the size of the worst-case solution, as a function of n. The following results are known:

Theorem 2. If P is a set of n points lying in the unit square, then

(i) $0.8474n^{1/2} \leqslant \operatorname{Sup}_P C_{GR}(P) \leqslant 1.074n^{1/2} + O(\log n)$ (GREEDY [4]),

(ii) $\operatorname{Sup}_P C_S(P) \leqslant 0.707n^{1/2} + O(1)$ (STRIP [23]),

(iii) $\operatorname{Sup}_P C_R(P) = 1.436n^{1/2} + O(1)$ (RECTANGLE [23]),

(iv) $0.932n^{1/2} \leqslant \operatorname{Sup}_P C_{SR}(P) \leqslant 1.014n^{1/2} + o(n^{1/2})$ (SPIRAL RACK [12]),

(v) $0.537n^{1/2} \leqslant \operatorname{Sup}_P C_{MIN}(P) \leqslant 0.707n^{1/2} + O(1)$.

The upper bound for the optimal solution given in (v) derives from the STRIP heuristic analyzed in [23]. The lower bound appears in several places [4, 13, 23] and is obtained by considering the regular hexagonal lattice which is known to give the densest packing of the plane by unit circles [21]. Consideration of this lattice also gives the lower bound for GREEDY in (i). The upper bound for GREEDY is obtained by first computing the largest possible minimum distance between a set of n points in the unit square. This again relates to several classical packing problems in the mathematical literature. GREEDY can of course be used in arbitrary dimensional spaces, and upper bounds for arbitrary dimensions can be obtained by using Blichfeldt's theorem for packing hyperspheres into the unit hypercube [4]. The upper bound for STRIP comes from the clever trick of superimposing the two partitions of the unit square and then by computing a simple bound on the length of the two tours combined. An averaging argument shows that there must be a matching of weight at most one-quarter of this total [23]. The analysis of RECTANGLE is much more complex, partly because the depth of the recursion is fixed in advance and also because the region is a rectangle, rather than the unit square, requiring additional arguments [24]. The results in (iv) concerning SPIRAL RACK are obtained by a completely different technique. The bound is obtained in a very compact and elegant manner by formulating and solving a small linear program. The method is quite general and is used in [13] to obtain bounds for many alternative linear heuristics in both the L_2 and L_∞ metrics. This bound was obtained with the tuning parameter α set at approximately 1.712, and holds both with and without the preprocessing step.

D. Average-Case Results

In this section we review some results on the expected size of the matchings produced by the various heuristics, when the input points are located uniformly in the

unit square. These results are interesting for two reasons. From a practical standpoint, the points in a dense drawing to be plotted are very likely to be approximately uniformly distributed; thus this measure may be more appropriate than the absolute worst-case bound. From a mathematical viewpoint, it is of independent interest to find the expected size of the optimal minimum-weight matching. It is here that the advantage of heuristics become apparent: it is virtually impossible to do an average-cost analysis on Edmonds' algorithm. Indeed, a version of the STRIP heuristic for the traveling salesman problem was devised in [6] to analyze the expected size of the optimal tour. Papadimitriou [17] extended these results for matchings. A short proof is contained in Steele [22]. The following theorem presents the known results for three of the heuristics: no results are known for GREEDY.

Theorem 3. If P is a set of n points distributed uniformly in the unit square, then

(i) $E(C_S(P)) = 0.474n^{1/2} + o(n^{1/2})$ (STRIP using only T_1 [23]),

(ii) $E(C_R(P)) = 0.516n^{1/2} + o(n^{1/2})$ (λ-RECTANGLE with $\lambda = [\log n] + 7$ [23]),

(iii) $E(C_{SR}(P)) = 0.484n^{1/2} + o(n^{1/2})$ (SPIRAL RACK, with $\alpha = 1.12$ and preprocessing [13]),

(iv) $E(C_{MIN}(P)) \leqslant 0.402n^{1/2} + o(n^{1/2})$ [(17)].

The bound (iv) for the optimal solution is obtained from a modified STRIP heuristic which also runs in $O(n \log n)$ time. Papadimitriou conjectures that $E(C_{MIN}(P)) = 0.35n^{1/2}$ and empirical studies indicate values close to this [13]. Probabilistic results for other matching heuristics are contained in [19] and [13]. Both papers also contain results for the L_∞ metric.

E. Numerical Results for a Practical Example

The most comprehensive numerical tests have been performed by Iri et al. and are reported in [13]. We reproduce some of their results in Table I. Some words of explanation are in order. The data is a road map of the Tokyo area with $n = 254$ points. The first row of the table indicates the plotter utilization when the edges are plotted as they appear; i.e., no attempt is made to find an Eulerian path. The second row indicates the results obtained by forming an Eulerian path by arbitrarily matching the vertices of odd degree, i.e., by using a random matching. The next three rows indicate the results obtained by using the SPIRAL RACK and STRIP heuristics. Unfortunately, no results are available for the GREEDY and RECTANGLE heuristics. In the column representing wasted pen movement, both the absolute waste and the waste normalized to the unit square and divided by $n^{1/2}$ are given. [13] also contains some extensive Monte Carlo experiments which were used to tune the various heuristics.

III. HEURISTICS FOR MATCHINGS WITH POSITIVE EDGE WEIGHTS

A. Description of Heuristics

The heuristics described in this section are designed to find matchings in the complete graph K_n or the complete bipartite graph $K_{n,n}$. The latter problem is better known as the *assignment problem* and is often stated in terms of an $n \times n$ cost *matrix*. The first published study of heuristics for the assignment problem appears in Kurtz-

TABLE I. An application: drawing the road map of the Tokyo city area; 850 edges, 513 vertices (including 254 vertices of odd degree) (courtesy Iri et al. [13]).

Algorithm		CPU time (ms)			Pen movement (cm)	
		For making a matching	For finding an Eulerian path	Total	Wasted with pen off[b]	For drawing real edges
Most primitive way of drawing (drawing edges as they are input)	L_2	0	0	0	1614(2.89)	896
	L_∞	0	0	0	1519(2.72)	817
Making pairs in the order of appearance	L_2	0.2	15	15	576(1.03)	896
	L_∞	0.2	15	15	566(1.01)	817
Serpentine (tour) No. 3	L_2	3.1	15	18	238(0.43)	896
	L_∞	2.3	15	18	206(0.37)	817
Spiral rack (preprocess, tour) No. 8	L_2	3.0	15	18	227(0.41)	896
	L_∞	1.9	15	17	214(0.39)	817
Strip No. 11	L_2	12	15	27	183(0.33)	896
	L_∞	9	15	24	164(0.30)	817
Exact[a] No. 12	L_2	34×10^3	15	34×10^3	128(0.23)	896
	L_∞	31×10^3	15	31×10^3	114(0.20)	817

[a]Coded in PASCAL; others are coded in VOS2/VOS3 optimizing FORTRAN 77 (OPT=2) on the HITAC M-200H.
[b]Length of the wasted pen movement scaled to a unit square divided by $n^{1/2}$.

berg's master's thesis, which was published in [15]. This paper contains four heuristics: the matrix scan method, the row scan method, the row/column scan method, and the decomposition method. The matrix scan method turns out to be the GREEDY heuristic introduced in Section II.A with very minor modifications to this context. This heuristic was also analyzed independently by the author for the complete graph K_n in [3], where it was called GREEDYII. Although Kurtzberg reports an $O(n^3)$ time complexity, this can be reduced to $O(n^2 \log n)$ by sorting. Kurtzberg's row/column scan heuristic uses the following variation of the greedy principle:

procedure VERTEX-SCAN
begin
while some vertex is unmatched
do begin
Pick some vertex i at random.
Choose edge (i,j) of minimum weight.
Delete i, j and all adjacent edges to both vertices.
end
end;

The row/column scan is in fact two calls to VERTEX-SCAN. In the first call, all vertices i chosen in the *while* loop are selected from the same part of the bipartite graph. In the second call, all vertices are chosen from the other part. (In terms of matrices, first the rows are scanned and then the columns; hence the name row/column scan). The minimum of the two matchings obtained is chosen. This heuristic is easily implemented in $O(n^2)$ time. The procedure VERTEX-SCAN can, of course, be applied to the complete graph K_n. It appears in this context in [3] where it is called GREEDYI.

Kurtzberg's decompose heuristic can best be described in terms of matrices. Let $D[1 \cdots n, 1 \cdots n]$ be the cost matrix corresponding to the weighted graph $K_{n,n}$. A parameter k, which is assumed to divide n, is fixed in advance.

procedure DECOMPOSE
begin
1. Partition D into k^2 submatrices R_{ij}, $1 \leqslant i, j \leqslant k$, of size $n/k \times n/k$.
2. Solve each R_{ij} using an optimal algorithm obtaining costs a_{ij} and permutation matrices Y_{ij}.
3. Solve $A = (a_{ij})$, $1 \leqslant i, j \leqslant k$, using an optimal algorithm obtaining permutation matrix $Q = (q_{ij})$.
4. Build the permutation matrix $M[1 \cdots n, 1 \cdots n]$ by inserting those blocks Y_{ij} for which $q_{ij} = 1$, and setting all other elements to zero.

end;

Kurtzberg does not include a time complexity analysis of DECOMPOSE, so we include one here. The heuristic requires the solution of k^2 assignment problems each of size $n/k \times n/k$, in Step 2. This takes time $O(n^3/k^3)$ using the Hungarian algorithm [18]. Step 3 requires the solution of one problem of size $k \times k$ taking $O(k^3)$ time. Steps 1

and 4 require $O(n^2)$ time each; hence the overall complexity is $O(n^2) + O(k^3) + O(n^3/k)$, which is minimized when $k = n^{3/4}$. Thus the overall complexity of DECOMPOSE is $O(n^{2.25})$. One motivation for using DECOMPOSE is the situation where the given matrix is too large to fit into main memory. Then setting $k = n^{1/2}$ ensures that no problem of size $n^{1/2} \times n^{1/2}$ need ever be considered; however, the overall complexity is $O(n^{2.5})$ for this value of k.

Donath [9] improves on the row/column scan method by introducing a multi-column scan heuristic. This heuristic involves successive investigation of the next smallest element in the columns being scanned, whenever the current element being considered is in a row which was assigned to another column in a previous step. The multiscan heuristic can be implemented in $O(n^2 \log n)$ time.

Finally, a new $O(n^2)$ heuristic for the assignment problem, SELECT, was analyzed by Lai in his master's thesis. This procedure involves selecting a linear number of low-cost edges from $K_{n,n}$ and then finding a maximum-cardinality matching on these edges. Fill in is used in case a perfect matching is not found. The success of the procedure depends on the choice of low-cost edges.

procedure SELECT
begin
1. Suppose n is even and partition the two parts of $K_{n,n}$ into S_1, S_2 and T_1, T_2 each with $\frac{1}{2}n$ vertices; fix $d \geqslant 5$.
2. For each $v \in S_1$ choose the d lowest-cost edges to T_1;
   for each $v \in S_2$ choose the d lowest-cost edges to T_2;
   for each $v \in T_1$ choose the d lowest-cost edges to S_2;
   for each $v \in T_2$ choose the d lowest-cost edges to S_1.
3. Find a maximum cardinality matching on the edges chosen in Step 2.
4. Use a greedy heuristic to complete the matching from Step 3 into a perfect matching.

end;

Figure 7 illustrates the selection process of Step 2. Note that at most $2dn$ edges are chosen. By fixing d (in practice $d = 5$ is chosen) the procedure SELECT can be

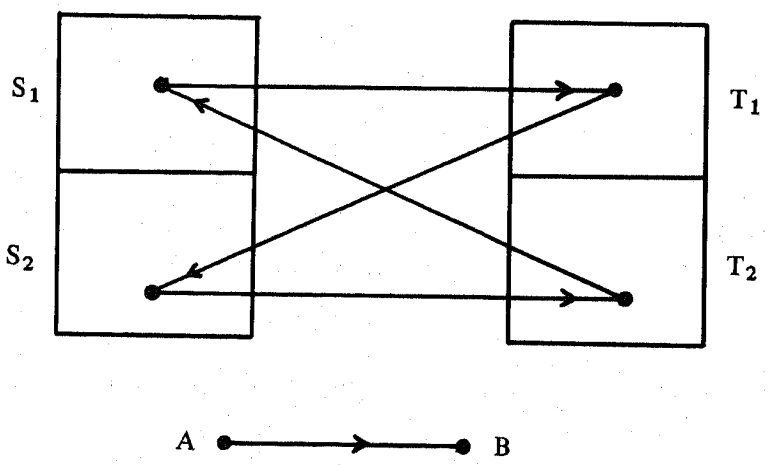

FIG. 7. Illustrating the second step of SELECT: for each vertex $v \in A$, choose the d smallest-weight edges into B.

implemented in $O(n^2)$ time. This heuristic was motivated by a nonconstructive technique of Walkup [27] used to obtain a remarkable upper bound on the expected optimal value of the assignment problem. This result is discussed in Section III.C.

This concludes our description of heuristics for the arbitrarily weighted matching problem. The remaining subsections contain results on the cost performance of these heuristics.

B. Worst-Case Results

Very few worst-case results are available for the minimum-weight matching problem with arbitrary positive edge weights, either with relative error bounds or with absolute error bounds. The reason for this is the fact that maximum-cardinality matching in an arbitrary graph $G = (V, E)$ can be embedded into this problem. If G has n vertices then G can be embedded into a weighted graph K_n by setting the weight on (i, j) to one if $(i, j) \in E$ and to some other constant N otherwise. Any heuristic that delivers a guaranteed bound on the relative or absolute error must therefore be powerful enough to find maximum-weight matchings in arbitrary graphs.

A ratio bound is, however, possible for the case of *maximum* weight matching. Let $C^*_{\mathrm{GR}}(G)$ be the weight of the matching obtained by GREEDY on an arbitrary weighted graph G with non-negative edge weights modified to pick the *heaviest* available edge. Let $C_{\mathrm{MAX}}(G)$ be the weight of the maximum-weight matching.

Theorem 4.

$$C_{\mathrm{MAX}}(G)/C^*_{\mathrm{GR}}(G) \leqslant 2.$$

Proof. Let x be the weight of the first edge (i, j) that is selected by GREEDY, so that x is in fact an edge of maximum weight in G. Now when (i, j) and all incident edges are deleted, at most two edges of the optimal matching may be removed. Further, the sum of their weights cannot exceed $2x$. The other $n - 2$ or more edges of the optimal matching are candidates for selection at the next iteration of GREEDY. The argument may be repeated for each of the first $\frac{1}{2}n$ iterations of GREEDYII. Since all edge weights are non-negative, the theorem is proved.

This result appears in [4] for the case where $G = K_n$; however, the proof holds for arbitrary G. Thus GREEDY (MATRIX-SCAN) produces a matching of at least half the weight of the maximum. A bound can also be found for DECOMPOSE modified in the obvious way to find maximum-weight matchings with weight $C^*_{\mathrm{D}}(K_{n,n})$.

Theorem 5.

$$C_{\mathrm{MAX}}(K_{n,n})/C^*_{\mathrm{D}}(K_{n,n}) \leqslant \min\{k, n/k\}.$$

Proof. (sketch). Referring to Step 3 of procedure DECOMPOSE, it is clear that $\Sigma_{i,j} a_{i,j} \geqslant C_{\mathrm{MAX}}(K_{n,n})$. Now $K_{n,n}$ can be decomposed into k independent and disjoint matchings, hence

$$C^*_{\mathrm{D}}(K_{n,n}) \geqslant \frac{\Sigma_{i,j} a_{i,j}}{k} \geqslant \frac{C_{\mathrm{MAX}}(K_{n,n})}{k}.$$

Now consider the matrix $A^* = (a^*_{i,j})$, $1 \leqslant i, j \leqslant k$, defined by

$$a^*_{i,j} = \begin{cases} a_{i,j}, & \text{if } R_{i,j} \text{ contains an element in optimum matching,} \\ 0, & \text{otherwise.} \end{cases}$$

Then A^* can have at most m non-negative entries in each row and column, and these non-negative entries can be decomposed into at most m perfect matchings. Further, $\Sigma_{i,j} a^*_{i,j} \geqslant C_{\text{MAX}}(K_{n,n})$, hence

$$C^*_{\text{D}}(K_{n,n}) \geqslant \frac{\Sigma_{i,j} a^*_{i,j}}{m} \geqslant \frac{C_{\text{MAX}}(K_{n,n})}{m},$$

which proves the theorem.

It can be shown that the bound in Theorem 5 is best possible; this is immediate for the boundary situations when $k = 1$ or n. Assuming an implementation of DECOMPOSE with $k = n^{3/4}$, we obtain a worst-case ratio bound of $n^{1/4}$. Examples can easily be constructed to show that no similar bounds are possible for the other two heuristics described in Section III.A.

C. Average-Case Results

Considerably more is known about average-case bounds for the weighted matching problem with non-negative edge weights. Let $\overline{K}_{n,n}$ ($\overline{K}_n$) represent a random weighted complete bipartite graph (complete graph), with edge weights chosen independently and uniformly on the interval [0, 1]. Let $C_{\text{GR}}(\overline{K}_{n,n})$, $C_{\text{RC}}(\overline{K}_{n,n})$, $C_{\text{D}}(\overline{K}_{n,n})$, $C_{\text{S}}(\overline{K}_{n,n})$, $C_{\text{MIN}}(\overline{K}_{n,n})$ be the costs obtained by the procedures GREEDY (MATRIX-SCAN), ROW/Column Scan, DECOMPOSE, SELECT, and by the optimum algorithm on $\overline{K}_{n,n}$. Let $C_{\text{GR}}(\overline{K}_{2n})$ and $C_{\text{VS}}(\overline{K}_{2n})$ be the costs obtained by GREEDY and VERTEX-SCAN on $\overline{K}_{2n}$. Finally, let $H_n = 1 + \frac{1}{2} + \cdots + 1/n$ be the nth harmonic number. Then the following theorem summarizes known results on the expected value of these costs.

Theorem 6.

(i) $$E(C_{\text{GR}}(\overline{K}_{n,n})) = H_n + O(1), \quad [15]$$

(ii) $$E(C_{\text{RC}}(\overline{K}_{n,n})) = H_n, \quad [15]$$

(iii) $$E(C_{\text{VS}}(\overline{K}_{2n})) = \tfrac{1}{2}H_n, \quad [3]$$

(iv) $$\tfrac{1}{4} \ln (n-1) - \tfrac{1}{4} \leqslant E(C_{\text{GR}}(\overline{K}_{2n})) \leqslant \tfrac{1}{2} \ln n + 1, \quad [3]$$

(v) $$n^{3/4} \leqslant E(C_{\text{D}}(\overline{K}_{n,n})) \leqslant 3n^{3/4}, \quad [5]$$

(vi) $$E(C_{\text{S}}(\overline{K}_{n,n})) \leqslant 6, \quad [16]$$

(vii) $$n/(n+1) \leqslant E(C_{\text{MIN}}(\overline{K}_{n,n})) \leqslant 3. \quad [27]$$

(lower bound from [15])

Result (vii) was obtained by Walkup using nonconstructive methods. Donath [9] gives empirical evidence that the multicolumn scan heuristic obtains an expected solution of

at most 1.6. This paper also gives an argument to show an upper bound of 2.37; however, there seem to be some difficulties with the proof [27]. Motivated by (vii), Lai has provided a constructive proof of Walkup's theorem [16]. He also reports computational experience with SELECT on random problems of the type described above. On problems of size up to 150 X 150, he reports that SELECT obtains the optimum solution at least two-thirds of the time. The average relative error was only 0.2%. On problems of size 50 X 50 and 150 X 150, SELECT ran respectively twice and five times as fast as the optimum algorithm, which was an implementation of the Hungarian primal-dual [18]. For the value of the optimum solution, Lai's results support Donath's findings of an upper bound of about 1.6.

IV. CONCLUSIONS AND OPEN PROBLEMS

This paper contains a survey of known results for matching heuristics in two areas: the Euclidean matching problem and the assignment problem. A brief summary of the results presented in Sections II and III are contained in Tables II and III, respectively. The STRIP and SPIRAL RACK heuristics appear to be good choices for the Euclidean problem. Both are straightforward to implement, with STRIP having the better worst-case bound and SPIRAL RACK having the better time bound. RECTANGLE is also a very fast heuristic, but is more complex to program and has a poorer worst-case performance. GREEDY seems to have little to compensate for its slow running time.

For the minimum-cost assignment problem, SELECT gives near optimum performance with $O(n^2)$ time complexity. On the other hand, the algorithm is rather com-

TABLE II. Summary of results for Euclidean matching of n points in the unit square.

Algorithm	Time complexity	Upper bound on worst case (unit square)	Relative error	Empirical results[b] (see Section II.E)	
				Weight	Time
Optimal[a]	$O(n^3)$	$0.707n^{1/2}$	1	$0.23n^{1/2}$	34,000
GREEDY	$O(n^{3/2} \log n)$	$1.0474n^{1/2}$	$\frac{4}{3}n^{\lg 1.5}$	...	...
STRIP	$O(n \log n)$	$0.707n^{1/2}$	...	$0.33n^{1/2}$	12
SPIRAL RACK	$O(n)$	$1.014n^{1/2}$	...	$0.41n^{1/2}$	3
RECTANGLE	$O(n)$	$1.436n^{1/2}$	...	...	...

[a]Coded in PASCAL; other heuristics coded in FORTRAN 77.
[b]Empirical results for Tokyo road map, $n = 254$.

TABLE III. Summary of results for n X n assignment problem.

Algorithm	Time complexity	Upper bound on expected cost (minimization)	Ratio of optimal to heuristic costs (maximization)
Hungarian Primal-Dual	$O(n^3)$	3	1
SELECT	$O(n^2)$	6	$+\infty$
Row/Col Scan	$O(n^2)$	H_n	$+\infty$
GREEDY	$O(n^2 \log n)$	$H_n + O(1)$	2
DECOMPOSE	$O(n^{9/4})$	$n^{3/4}$	$n^{1/4}$

plex to program. For an easily programmable heuristic, Row/Column Scan appears to be a good choice. For the maximization problem, GREEDY (MATRIX-SCAN) has the advantage of a guaranteed bound on the relative error. DECOMPOSE has poor performance on all counts.

We conclude with some open problems.

(i) What is the worst-case and expected value of the minimum-cost matching on n points in the unit square?

(ii) What is the expected weight of the GREEDY matching on n points in the unit square?

(iii) Find an optimal implementation of GREEDY for either the Euclidean or the Assignment problem.

(iv) Find $O(n)$ or $O(n \log n)$ heuristics for the Euclidean problem with good ratio error bounds.

The collection of the material for this survey was greatly facilitated by the enthusiastic cooperation of several people. In particular, I would like to thank Ed Reingold for many stimulating correspondences on the subject matter of this survey. I am also most grateful to Masao Iri and Kazuo Murota of the University of Tokyo for many useful materials, including photographs of their figures. In this connection I would also like to thank Takao Nishizeki for helping bring our groups together. This work was supported by NSERC Grant No. A3013 and FCAC Grant No. EQ-1678.

References

[1] S. Akl, A statistical analysis of various aspects of the travelling salesman problem. Ph.D. Thesis, McGill University, 1978.

[2] S. Akl, A note on Euclidean matchings, triangulations and spanning trees. Unpublished.

[3] D. Avis, Two greedy heuristics for the weighted matching problem. Congr. Numer. XXI (1978) 65–76.

[4] D. Avis, Worst case bounds for the Euclidean matching problem. *Comput. Math. Appl.* 7 (1981) 251–257.

[5] D. Avis and L. Devroye, An analysis of a decomposition heuristic for the assignment problem. Unpublished.

[6] J. Beardwood, J. H. Halton, and J. M. Hammersley, The shortest path through many points. *Proc. Cambridge Philos. Soc.* 55 (1959) 299–327.

[7] J. L. Bentley and J. B. Saxe, Decomposable searching problems 1: Static to dynamic transformations. *J. Algorithms* 1 (1980) 301–358.

[8] N. Christofides, Worst case analysis of a new heuristic for the travelling salesman problem. Technical Report, G.S.I.A., Carnegie-Mellon University, 1976.

[9] W. E. Donath, Algorithm and average-value bounds for assignment problems. *IBM J. Res. Dev.* XIII (1969) 380–386.

[10] J. Edmonds, Paths, trees and flowers. *Canad. J. Math.* 17 (1965) 449–467.

[11] M. Iri and A. Taguchi, The determination of the pen movement of an XY-plotter and its computational complexity (in Japanese). *Proc. Spring Conf. of the O.R. Society of Japan*, P-8, 1980, 204–205.

[12] M. Iri, K. Murota, and S. Matsui, Linear-Time approximation algorithms for finding the minimum weight perfect matching on a plane. *Information Processing Lett.* 12 (1981) 206–209.

[13] M. Iri, K. Murota, and S. Matsui, Linear-time heuristics for the minimum weight perfect matching on a plane with an application to the plotter algorithm. Re-

search Memorandum RMI 81-07, revised November 1981, University of Tokyo. (An abridged version will appear in *Networks* 13 (1983) 67–92.

[14] R. M. Karp, A patching algorithm for the non-symmetric travelling salesman problem. Technical Report UCB/ERL/M78/2, Electronics Research Lab., University of California, Berkeley, 1978.

[15] J. M. Kurtzberg, On approximation methods for the assignment problems. *J. Assoc. Comput. Mach.* 9 (1962) 419–439.

[16] C. W. Lai, A heuristic for the assignment problem and related bounds. Technical Report 81.20, McGill University, 1981.

[17] C. H. Papadimitriou, The probabilistic analysis of matching heuristics. *Proc. 15th Allerton Conference on Communication Control and Computing*, 1977, pp. 368–378.

[18] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, Englewood Cliffs, NJ, 1982.

[19] E. M. Reingold and K. J. Supowit, Probabilistic analysis of divide and conquer heuristics for minimum weighted Euclidean matching. *Networks*. To appear.

[20] E. M. Reingold and R. E. Tarjan, On a greedy heuristic for complete matching. *SIAM J. Comput.* 10 (1981) 676–681.

[21] C. A. Rogers, *Packing and Covering*, Cambridge University, Cambridge, 1964.

[22] J. M. Steele, Subadditive Euclidean functionals and non-linear growth in geometric probability. *Ann. Probability* (1981) 365–376.

[23] K. J. Supowit, D. A. Plaisted, and E. M. Reingold, Heuristics for weighted perfect matching. *Proc. 12th Annual ACM Symp. on Theory of Computing*, 1980, pp. 398–419.

[24] K. J. Supowit and E. M. Reingold, Divide and conquer heuristics for minimum weighted Euclidean matching. Unpublished.

[25] K. J. Supowit, E. M. Reingold, and D. A. Plaisted, The travelling salesman problem and minimum matching in the unit square. *SIAM J. Comput.* To appear.

[26] D. W. Walkup, Matching in random regular bipartite graphs. *Discrete Math.* 31 (1980) 59–64.

[27] D. W. Walkup, On the expected value of a random assignment problem. *SIAM J. Comput.* 8 (1979), 440–442.

Received February 13, 1982
Accepted May 4, 1983