Heuristics for Planar Minimum-Weight Perfect Matchings

Masao Iri, Kazuo Murota, and Shouichi Matsui
Department of Mathematical Engineering and Instrumentation Physics, University of Tokyo, Japan

Several linear-time approximation algorithms for the minimum-weight perfect matching in a plane are proposed, and their worst- and average-case behaviors are analyzed theoretically as well as experimentally. A linear-time approximation algorithm, named the "spiral-rack algorithm (with preprocess and with tour)," is recommended for practical purposes. This algorithm is successfully applied to the drawing of road maps such as that of the Tokyo city area.

I. INTRODUCTION

Consider n (an even number) points in a plane. The problem of finding the minimum-weight perfect matching, i.e., determining how to match the n points in pairs so as to minimize the sum of the distances between the matched points, as well as Euler's problem of unicursal traversing on a graph, is of fundamental importance for optimizing the sequence of drawing lines by a mechanical plotter ([2-5, 8]; details are discussed in Sec. V).

The algorithm which exactly solves this problem in $O(n^3)$ time [6] seems to be too complicated from the practical point of view. Even approximation algorithms of $O(n^2)$ or $O(n \log n)$ [10] would not be satisfactory or need some improvement for the application to real-world problems of a size, say, n greater than 10^4. In contrast with the matching problem, an Eulerian path can be found in linear time in the number of edges.

In this paper, linear-time* approximation algorithms are proposed for the matching problem in a unit square; their worst-case performances are analyzed theoretically; their average-case performances are investigated both theoretically and experimentally for the case where n points are uniformly distributed on the unit square; and an application to the drawing of a road map is shown. The quality of an approximate solution is measured by the *absolute cost* of the matching, i.e., the sum of the distances

*We adopt the RAM model of computation which executes an arithmetic operation such as addition, multiplication, or integer division (hence, the "floor" operation) in a unit time [1].

CCC 0028-3045/83/010067-26$03.60

between the paired points, and not by the ratio of the cost to that of the exact optimal solution. For the distance, not only the L_2 distance (Euclidean distance) but also the L_∞ distance (maximum norm) is considered, the latter being more appropriate when the running time of a mechanical plotter is in question.

II. ALGORITHMS

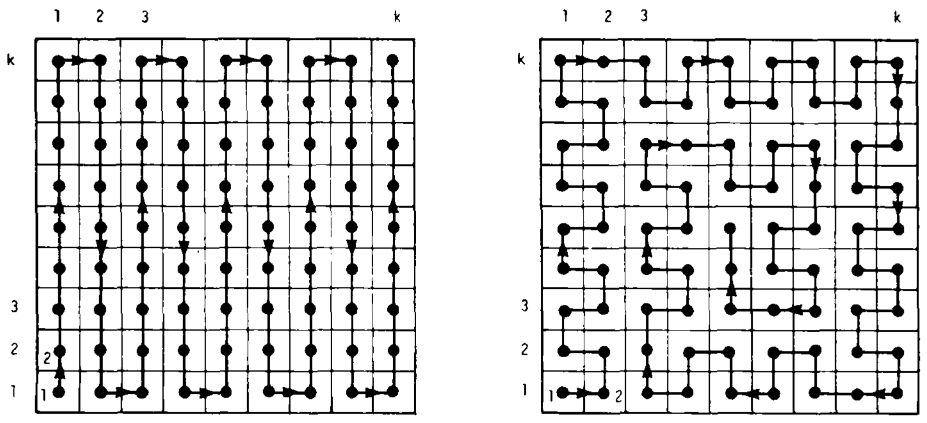
A. Straightforward Algorithms

We partition the unit square into k^2 square cells by dividing each side into k equal parts. Each point belongs to one of the k^2 cells. We determine the cell to which a point belongs by multiplying the coordinates (abscissa and ordinate) of the point by k and then truncating off the fractional parts. The cells are ordered in a prescribed order (see below). We number the n points to form a sequence which is consistent with the order of the cells they belong to; i.e., points in one and the same cell may arbitrarily be ordered among themselves, but points in different cells must be ordered consistently with the order of the cells. For the approximate solution we adopt the matching consisting of pairs of the $(2i-1)$st point and the $(2i)$th ($i = 1, 2, \ldots, \frac{1}{2}n$). Two types of cell orders are considered here—*serpentine* and *spiral rack* (see Fig. 1). The straightforward algorithm with the serpentine order will be called the *serpentine algorithm*, and that with the spiral rack order, the *spiral-rack algorithm.*

The following variants of the algorithms are also considered:

(i) *Preprocess.* First arbitrarily match as many pairs of points as possible within each cell separately, and then apply one of the above algorithms to the remaining points.

(ii) *Tour.* Make the matching $\{(2i, 2i+1) | i = 1, 2, \ldots, \frac{1}{2}n, (n+1 \equiv 1)\}$ in addition to the matching $\{(2i-1, 2i)\}$, and adopt the one with less cost for the solution.*



(i) serpentine cell order (ii) spiral rack cell order (k: odd)

FIG. 1. Typical cell orders.

*The idea of constructing a tour and taking the shorter is found in [10]. The basic idea of straightforward algorithms with tour is proposed in [2].

Provided k is taken as large in order as $n^{1/2}$, it is evident that the complexity of these algorithms is $O(n)$ both in time and in space.

In the following, the *cell distance* of two cells with respect to a prescribed cell order will mean the difference of the cell numbers in that ordering; e.g., two consecutive cells are at cell distance 1.

B. Bottom-Up Four-Square Algorithm

As above, we divide the unit square into k^2 cells of size $1/k$, where k is taken to be a power of 2, say, $k = 2^m$. After determining which point belongs to which cell, we match the points in pairs as far as possible within the cells. Then, we aggregate the neighboring four cells into a cell of size $2/k$, to have $\frac{1}{4}k^2$ larger square cells. Within each of these larger cells, we match the remaining points as far as possible. We repeat a similar process m times. This algorithm, with $k = O(n^{1/2})$, can be implemented to run in $O(n)$ time by the use of carefully designed data structure. {The four-square algorithm has been proposed in [10]. The simple-minded top-down recursion for it would seem to require $O(n \log n)$ time.}

C. Strip-with-Bucket Algorithm

We apply the serpentine algorithm (with tour and without preprocess) once to the original pattern and then to the pattern shifted by $1/2k$ horizontally. We adopt the matching with less cost for the solution. This algorithm may be viewed as a linear-time variant of the strip algorithm in [10] in which the sorting, whose time complexity is $O(n \log n)$, is approximated by the distribution into buckets.

III. WORST-CASE ANALYSIS

For a fixed algorithm with k^2 cells, let $\hat{M}_n(k)$ be the supremum of the costs of solutions over all possible configurations of n points in the unit square, and put

$$\hat{\mu} = \hat{\mu}(\alpha) = \lim_{n \to \infty} \hat{M}_n(\alpha n^{1/2})/n^{1/2}, \qquad \hat{\mu}_0 = \hat{\mu}(\hat{\alpha}_0) = \min_{\alpha} \hat{\mu}(\alpha).$$

The following analysis shows that the spiral-rack algorithm with tour is the best among the linear-time algorithms considered here so long as the worst-case behavior is concerned.

A. Straightforward Algorithms

An upper bound for $\hat{M}_n(k)$ of a straightforward algorithm can be obtained by means of linear programming.* The primal variable n_j denotes the number of edges in a matching (or in a tour in the analysis of the algorithms with tour) connecting points in two cells at cell distance $j - 1$; in particular, n_1 is the number of pairs within the same cells.

*A referee kindly informed the authors that the idea of using the duality of linear programs was also thought of by Papadimitriou (unpublished) to analyze the minimum-spanning-tree heuristics described in [10].

We shall first illustrate the idea of using linear programs for the analysis of the serpentine algorithm without preprocess and without tour for the L_∞ distance. Let us consider the

Primal program:

$$\text{maximize} \qquad f \equiv \sum_{j=1} c_j n_j$$

$$\text{subject to} \qquad \sum_{j=1} n_j = \frac{n}{2},$$

$$\sum_{j=1} (j-1) n_j \leqslant k^2 - 1,$$

$$n_j \geqslant 0,$$

where $c_j = j/k$ is an upper bound for the L_∞ distance between two points contained in two cells at cell distance $j-1$. The first constraint comes from the fact that the total number of pairs in a perfect matching is one-half of the number of points, and the second from the fact that the cell distance between the "first" point and the "last," i.e., the left-hand side, does not exceed the cell distance between the first cell and the last. Obviously, the value $\hat{f}$ of the objective function for the optimal solution of the primal program gives an upper bound for $\hat{M}_n(k)$.

The dual program is then

Dual program:

$$\text{minimize} \qquad g \equiv \tfrac{1}{2} n x + (k^2 - 1) y,$$

$$\text{subject to} \qquad x + (j-1) y \geqslant c_j \quad (j = 1, 2, \ldots),$$

$$y \geqslant 0.$$

As is well known, the value g of the dual objective function for any feasible solution of the dual program is not smaller than $\hat{f}$, i.e.,

$$\hat{M}_n(k) \leqslant \hat{f} \leqslant g,$$

so that g can be used as an upper bound for $\hat{M}_n(k)$. In particular, the minimum value $\hat{g}$ of the dual program gives a good upper bound, which often, as in this case, turns out to be asymptotically tight.

The optimal solution of the dual program is easily shown to be $(\hat{x}, \hat{y}) = (1/k, 1/k)$, for which $\hat{x} + (j-1)\hat{y} = c_j$ for all $j = 1, 2, \ldots$, and $\hat{g} = n/2k + k - 1/k$. Hence we have

$$\hat{M}_n(k) \leqslant n/2k + k - 1/k. \tag{1}$$

The optimal solution of the primal program is degenerate; for example, we have

$$\hat{n}_1 = \tfrac{1}{2} n - k - 1, \qquad \hat{n}_k = k + 1,$$

$$\hat{n}_j = 0 \text{ for the other } j,$$

which gives

$$\hat{f} = n/2k + k - 1/k = \hat{g}. \tag{2}$$

Since the variables n_j represent the number of points so that they are to be integers, there is not in general a distribution of n points in the square such that the serpentine algorithm (without preprocess and without tour) will yield a matching with the cost equal to (2). However, it is readily seen that there is a distribution (Fig. 2) for which the algorithm will yield a matching with cost asymptotically equal to (2). Therefore, the upper bound (2) is asymptotically attainable; i.e., we have

$$\hat{\mu}(\alpha) = 1/2\alpha + \alpha,$$

hence

$$\hat{\mu}_0 = \sqrt{2} \quad \text{for} \quad \hat{\alpha}_0 = 1/\sqrt{2}.$$

The worst-case costs of the other straightforward algorithms and their variants can be analyzed by slightly modifying the constraints and/or the expressions for c_j, as is sketched below, where, for the sake of simplicity, we shall consider asymptotic properties only, e.g., we shall replace the right-hand side $k^2 - 1$ of the constraint of the primal program by k^2. It is interesting to note that, in the worst-case analysis, a straightforward algorithm with tour and without preprocess has the same linear program as the corresponding algorithm with tour and with preprocess. The results are summarized in Table I.

1. Serpentine Algorithm

For the serpentine cell order, we have

L_2 distance: $c_j = (1 + j^2)^{1/2}/k \quad (j = 1, 2, \ldots),$

L_∞ distance: $c_j = j/k \quad (j = 1, 2, \ldots).$

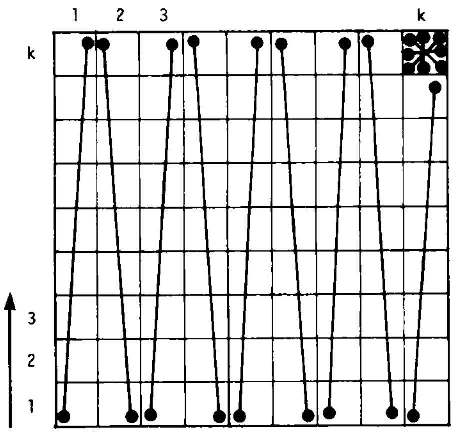


FIG. 2. Worst configuration for the serpentine algorithm without tour.

TABLE I. Asymptotic supremum $\hat{\mu}_0$ and expectation μ_0 of the cost/$n^{1/2}$ for the optimal value $\hat{\alpha}_0$ and α_0 of the parameter $\alpha = k/n^{1/2}$

Algorithm			L_2 Distance				L_∞ Distance			
Number	Name	Complexity	$\hat{\mu}_0$	$\hat{\alpha}_0$	μ_0	α_0	$\hat{\mu}_0$	$\hat{\alpha}_0$	μ_0	α_0
1	Serpentine	$O(n)$	1.682	0.841	0.585	0.77	1.414	0.707	0.545	0.73
2	Serpentine (preprocess)	$O(n)$	1.682	0.841	0.637	0.79	1.414	0.707	0.603	0.75
3	Serpentine (tour)	$O(n)$	1.189	1.189	0.585[a]	0.77[a]	1	1	0.545[a]	0.73[a]
4	Serpentine (preprocess, tour)	$O(n)$	1.189	1.189	0.637[a]	0.79[a]	1	1	0.603[a]	0.75[a]
5	Spiral-rack	$O(n)$	$\leqslant 1.434$ $\geqslant 1.318$	[1.211][b]	0.495	1.21	1.225	1.225	0.450	1.17
6	Spiral-rack (preprocess)	$O(n)$	1.245	1.136	0.484	1.12	1	1	0.443	1.07
7	Spiral-rack (tour)	$O(n)$	$\leqslant 1.014$ $\geqslant 0.932$	[1.712][b]	0.495[a]	1.21[a]	0.866	1.732	0.450[a]	1.17[a]
8	Spiral-rack (preprocess, tour)	$O(n)$	$\leqslant 1.014$ $\geqslant 0.932$	[1.712][b]	0.484[a]	1.12[a]	0.866	1.732	0.443[a]	1.07[a]
9	Bottom-up four-square	$O(n)$	$\leqslant 1.936$ $\geqslant 1.789$	[0.548 ~1.095][b,c]	0.550[d]	2.78[d]	$\leqslant 1.369$ $\geqslant 1.265$	[0.548 ~1.095][b,c]	0.492[d]	2.78[d]
10	Strip-with-bucket	$O(n)$	$\leqslant 1.189$ $\geqslant 1.057$	[1.189][b]	0.585[a]	0.77[a]	1	1	0.545[a]	0.73[a]
11	Strip [10]	$O(n \log n)$	0.707[e]	. . .	0.474[a,f]	. . .	. . .	. . .	0.436[a,f]	. . .
12	Exact	$O(n^3)$	$\geqslant 0.537$[e]	. . .	0.35[g]	. . .	$\geqslant 0.5$	. . .	. . .	. . .

[a]It is assumed that $M_n/n^{1/2}$ converges in probability to a constant for the serpentine and the spiral-rack algorithms.
[b]The value corresponding to an upper bound for $\hat{\mu}_0$, which is not asymptotically tight.
[c]The optimal value of α should be determined in this range in such a way that $k = \alpha n^{1/2}$ may be a power of 2.
[d,f]By elementary probability calculation. See [4] for detail of f.
[e]From [10].
[g]This is the value conjectured in [7]. However, according to our experiment, this value seems to lie between 0.32 and 0.33.

The linear programs for the four variants of the serpentine algorithm as well as the resulting upper bounds are as follows.

a. Without Preprocess and Without Tour[*]

Primal program:

maximize $$f \equiv \sum_{j=1} c_j n_j$$

subject to $$\sum_{j=1} n_j = \frac{n}{2},$$

$$\sum_{j=1} (j-1) n_j \leqslant k^2,$$

$$n_j \geqslant 0.$$

Dual program:

minimize $$g \equiv \tfrac{1}{2} n x + k^2 y$$

subject to $$x + (j-1) y \geqslant c_j \quad (j = 1, 2, \ldots)$$

$$y \geqslant 0.$$

L_2 distance: $\hat{\mu}(\alpha) = 1/\sqrt{2}\alpha + \alpha$, $\hat{\mu}_0 = 2^{3/4} \risingdotseq 1.682$, $\hat{\alpha}_0 = 2^{-1/4} \risingdotseq 0.841$;

L_∞ distance: $\hat{\mu}(\alpha) = 1/2\alpha + \alpha$, $\hat{\mu}_0 = \sqrt{2} \risingdotseq 1.414$, $\hat{\alpha}_0 = 1/\sqrt{2} \risingdotseq 0.707$.

The upper bound in either distance is asymptotically attained by the configuration given in Figure 2, where we have k pairs at L_2 (or L_∞) distance 1, i.e., $n_k = k$ and $\frac{1}{2}n - k$ pairs at L_2 distance $\sqrt{2}/k$ (or L_∞ distance $1/k$), i.e., $n_1 = \frac{1}{2}n - k$.

b. With Preprocess and Without Tour

Primal program:

maximize $$f \equiv \sum_{j=1} c_j n_j$$

subject to $$\sum_{j=1} n_j = \tfrac{1}{2} n,$$

$$\sum_{j=2} j n_j \leqslant k^2,$$

$$n_j \geqslant 0.$$

Dual program:

minimize $$g \equiv \tfrac{1}{2} n x + k^2 y$$

*The case for the L_∞ distance is already explained above.

subject to
$$x \geqslant c_1,$$
$$x + jy \geqslant c_j \quad (j = 2, 3, \ldots),$$
$$y \geqslant 0.$$

Here it should be noted that, owing to preprocessing, the first point of a pair in a matching cannot lie in the same cell that the second point of the preceding pair lies in, so that we have the coefficient j, instead of $j - 1$, in the inequality constraint of the primal program.

L_2 distance: $\hat{\mu}(\alpha) = 1/\sqrt{2}\alpha + \alpha$, $\hat{\mu}_0 = 2^{3/4} \fallingdotseq 1.682$, $\hat{\alpha}_0 = 2^{-1/4} \fallingdotseq 0.841$;

L_∞ distance: $\hat{\mu}(\alpha) = 1/2\alpha + \alpha$, $\hat{\mu}_0 = \sqrt{2} \fallingdotseq 1.414$, $\hat{\alpha}_0 = 1/\sqrt{2} \fallingdotseq 0.707$.

The bound in either distance is asymptotically attainable again by the configuration of Figure 2.

c. With/Without Preprocess and With Tour

Primal program:

maximize
$$f \equiv \sum_{j=1} \frac{c_j n_j}{2}$$

subject to
$$\sum_{j=1} n_j = n,$$
$$\sum_{j=1} (j-1) n_j \leqslant k^2,$$
$$n_j \geqslant 0.$$

Dual program:

minimize
$$g \equiv nx + k^2 y$$

subject to
$$x + (j-1)y \geqslant \tfrac{1}{2} c_j \quad (j = 1, 2, \ldots),$$
$$y \geqslant 0.$$

Here, n_j denotes the number of edges whose vertices are at cell distance $j - 1$ in a tour. Obviously, one-half of the tour length gives an upper bound for the better matching in the tour. Note also that the coefficient $j - 1$ in the inequality constraint of the primal program here remains unaltered for the case with preprocess.

L_2 distance: $\hat{\mu}(\alpha) = 1/\sqrt{2}\alpha + \frac{1}{2}\alpha$, $\hat{\mu}_0 = 2^{1/4} \fallingdotseq 1.189$, $\hat{\alpha}_0 = 2^{1/4} \fallingdotseq 1.189$;

L_∞ distance: $\hat{\mu}(\alpha) = 1/2\alpha + \frac{1}{2}\alpha$, $\hat{\mu}_0 = 1$, $\hat{\alpha}_0 = 1$.

The configuration of Figure 3 asymptotically attains the upper bound in either case.

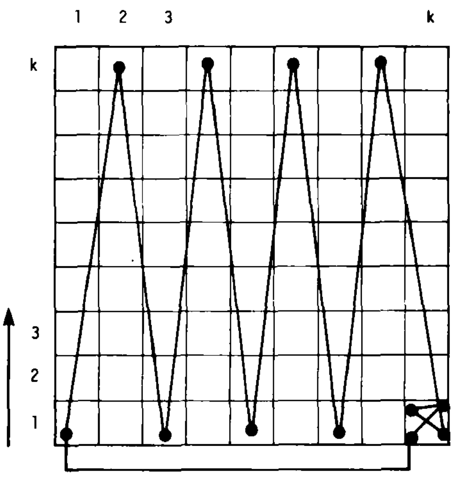


FIG. 3. Worst configuration for the serpentine algorithm with tour.

2. Spiral-Rack Algorithm

Upper bounds for $\hat{M}_n(k)$ of the spiral-rack algorithm which are obtained from the three linear programs outlined above with appropriately defined values c_j, are not asymptotically tight. Therefore we should resort to a more elaborate linear programming formulation.

Consider, for example, the edges connecting points in two cells at cell distance 2 with respect to the spiral-rack cell order. Some of those edges can have bound $3/k$ in L_∞ length, but the L_∞ distance for the rest of them can be at most $2/k$. With this observation, we introduce c_j' $(j = 3, 5, 7, \ldots)$ in addition to c_j. The former takes care of exceptions and the latter is an upper bound valid for the majority. We set

L_2 distance: $c_1 = \sqrt{2}/k,$

$$c_j = \begin{cases} \{1 + [\frac{1}{2}(j+2)]^2\}^{1/2}/k & (j = 2, 6, 10, \ldots), \\ \{4 + [\frac{1}{2}(j+1)]^2\}^{1/2}/k & (j = 3, 7, 11, \ldots), \\ \{4 + [\frac{1}{2}(j+2)]^2\}^{1/2}/k & (j = 4, 8, 12, \ldots), \\ \{1 + [\frac{1}{2}(j+1)]^2\}^{1/2}/k & (j = 5, 9, 13, \ldots), \end{cases}$$

$$c_j' = \begin{cases} \{1 + [\frac{1}{2}(j+3)]^2\}^{1/2}/k & (j = 3, 7, 11, \ldots), \\ \{4 + [\frac{1}{2}(j+3)]^2\}^{1/2}/k & (j = 5, 9, 13, \ldots), \end{cases}$$

L_∞ distance: $c_j = \lfloor \frac{1}{2}(j+2) \rfloor /k \quad (j = 1, 2, 3, \ldots),$

$$c_j' = \tfrac{1}{2}(j+3)/k \qquad (j = 3, 5, 7, \ldots),$$

where $\lfloor x \rfloor$ denotes the largest integer not exceeding x.

The modified linear programs, with variables n_j $(j = 1, 2, 3, \ldots)$ and n_j' $(j = 3, 5, 7, \ldots)$, for the four variants of the spiral-rack algorithm, are shown below, together with the resulting upper bounds for $\hat{\mu}(\alpha)$. In the following, Σ^o denotes summation over odd integers.

a. Without Preprocess and Without Tour

Primal program:

$$\text{maximize} \qquad f \equiv \sum_{j=1} c_j n_j + \sum_{j=3}^{o} c'_j n'_j$$

$$\text{subject to} \qquad \sum_{j=1} n_j + \sum_{j=3}^{o} n'_j = \frac{n}{2},$$

$$\sum_{j=1} (j-1) n_j + \sum_{j=3}^{o} (j-1) n'_j \leqslant k^2,$$

$$\sum_{j=3}^{o} n'_j \leqslant k,$$

$$n_j \geqslant 0,$$

$$n'_j \geqslant 0.$$

Dual program:

$$\text{minimize} \qquad g \equiv \tfrac{1}{2} n x + k^2 y + kz$$

$$\begin{aligned} \text{subject to} \qquad x + (j-1)y &\geqslant c_j \qquad (j = 1, 2, 3, \ldots), \\ x + (j-1)y + z &\geqslant c'_j \qquad (j = 3, 5, 7, \ldots), \\ y &\geqslant 0, \\ z &\geqslant 0. \end{aligned}$$

L_2 distance:

$$\hat{\mu}(\alpha) \begin{cases} = \min\left(\dfrac{\sqrt{2}}{2\alpha} + (\sqrt{5} - \sqrt{2})\alpha, \dfrac{3\sqrt{5} - \sqrt{13}}{4\alpha} + \dfrac{(\sqrt{13} - \sqrt{5})\alpha}{2}\right), & \alpha \leqslant 1, \\ \leqslant \min\left(\dfrac{3\sqrt{5} - \sqrt{13}}{4\alpha} + \dfrac{(\sqrt{13} - \sqrt{5})\alpha}{2}, \dfrac{2\sqrt{13} - 3}{4\alpha} + \dfrac{\alpha}{2}\right), & \alpha > 1. \end{cases}$$

Though the upper bound for $\alpha > 1$ is not tight, we have the inequality

$$\hat{\mu}_0 \leqslant (\sqrt{13} - \tfrac{3}{2})^{1/2} \fallingdotseq 1.451.$$

A sharper bound

$$\hat{\mu}_0 \leqslant [(2\sqrt{5} - 1)(\sqrt{13} + 1 - \sqrt{5})]^{1/2}/2 \fallingdotseq 1.434$$

may be obtained by refining the linear program in a way similar to that outlined later in Subsection *d*. We have also a lower bound

$$\hat{\mu}_0 \geqslant (\sqrt{5} - \tfrac{1}{2})^{1/2} \doteqdot 1.318$$

which is asymptotically realized by a configuration with

$$n_2 \approx [(j_o - 2\alpha^2 - 1)/2(j_o - 2)]n, \quad n_{j_o} \approx [(2\alpha^2 - 1)/2(j_o - 2)]n,$$
$$n_j = 0 \quad (j \neq 2, j_o),$$

where j_o is a sufficiently large integer.

L_∞ distance:

$$\hat{\mu}(\alpha) = \min\left(\frac{1}{2\alpha} + \alpha, \frac{3}{4\alpha} + \frac{\alpha}{2}\right), \quad \hat{\mu}_0 = \sqrt{\tfrac{3}{2}} \doteqdot 1.225, \quad \hat{\alpha}_0 = \sqrt{\tfrac{3}{2}} \doteqdot 1.225.$$

The supremum $\hat{\mu}(\alpha)$ in the L_∞ distance is asymptotically attained by a configuration with

$$\hat{n}_1 \approx (\tfrac{1}{2} - \alpha^2)n, \quad \hat{n}_2 \approx \alpha^2 n, \quad \hat{n}_j = 0 \quad (j \neq 1, 2) \text{ for } \alpha \leqslant 1/\sqrt{2};$$
$$\hat{n}_2 \approx [(j_o - 2\alpha^2 - 1)/2(j_o - 2)]n, \quad \hat{n}_{j_o} \approx [(2\alpha^2 - 1)/2(j_o - 2)]n,$$
$$\hat{n}_j = 0 \quad (j \neq 2, j_o) \text{ for } \alpha \geqslant 1/\sqrt{2},$$

where j_o is an even integer greater than $4\alpha^2$.

b. With Preprocess and Without Tour

Primal program:

maximize
$$f \equiv \sum_{j=1} c_j n_j + \sum_{j=3}^{o} c_j' n_j'$$

subject to
$$\sum_{j=1} n_j + \sum_{j=3}^{o} n_j' = \frac{n}{2},$$
$$\sum_{j=2} j n_j + \sum_{j=3}^{o} j n_j' \leqslant k^2,$$
$$\sum_{j=3}^{o} n_j' \leqslant k,$$
$$n_j \geqslant 0,$$
$$n_j' \geqslant 0.$$

Dual program:

$$\begin{array}{lll} \text{minimize} & g \equiv \frac{1}{2}n\,x + k^2 y + kz & \\ \text{subject to} & x \geqslant c_1, & \\ & x + jy \geqslant c_j & (j = 2, 3, \ldots), \\ & x + jy + z \geqslant c_j' & (j = 3, 5, \ldots), \\ & y \geqslant 0, & \\ & z \geqslant 0. & \end{array}$$

L_2 distance:

$$\hat{\mu}(\alpha) = \min\left(\frac{1}{\sqrt{2}\alpha} + \frac{(\sqrt{13} - \sqrt{2})\alpha}{4}, \frac{\sqrt{13} - 2}{2\alpha} + \frac{\alpha}{2}\right),$$

$$\hat{\mu}_0 = (\sqrt{13/2} - 1)^{1/2} \doteqdot 1.245, \quad \hat{\alpha}_0 = 2/(\sqrt{26} - 2)^{1/2} \doteqdot 1.136.$$

For $\alpha \leqslant \sqrt{2}$, the asymptotic supremum $\hat{\mu}(\alpha)$ in the L_2 distance is attained by a configuration with

$$\hat{n}_1 \approx \tfrac{1}{4}(2 - \alpha^2)n, \quad \hat{n}_4 \approx \tfrac{1}{4}\alpha^2 n, \quad \hat{n}_j = 0 \quad (j \neq 1, 4).$$

For $\alpha \geqslant \sqrt{2}$, the constraints for $j = 4$ and $j = \infty$ in the dual program are active. The supremum $\hat{\mu}(\alpha)$ is attained by the configuration shown in Figure 4, where the shaded core, consisting of $2n - 4(\alpha - \sqrt{2})n^{1/2}$ cells, contains $\hat{n}_4 \approx \frac{1}{2}n - (\alpha - \sqrt{2})n^{1/2}$ pairs at L_2 distance $\sqrt{13}/k$ and the outer shell contains $\hat{n}_\infty \approx (\alpha - \sqrt{2})n^{1/2}$ pairs at L_2 distance $1 - i/k$ $[i = 1, 2, \ldots, (\alpha - \sqrt{2})n^{1/2}]$, the total length in the outer shell amounting to $(\frac{1}{2}\alpha - 1/\alpha)n^{1/2}$.

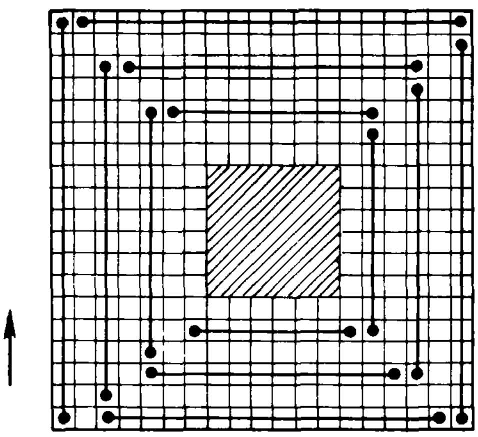


FIG. 4. Worst configuration for the spiral-rack algorithm with preprocess and without tour in the L_2 distance.

L_∞ distance: $\hat{\mu}(\alpha) = 1/2\alpha + \frac{1}{2}\alpha, \quad \hat{\mu}_0 = 1, \quad \hat{\alpha}_0 = 1.$

The asymptotic supremum $\hat{\mu}(\alpha)$ in the L_∞ distance is attained by a configuration with

$$\hat{n}_1 \approx [(j_o - 2\alpha^2)/2j_o]n, \quad \hat{n}_{j_o} \approx (\alpha^2/j_o)n, \quad \hat{n}_j = 0 \quad (j \neq 1, j_o),$$

where j_o is an even integer greater than $2\alpha^2$.

c. With/Without Preprocess and With Tour

Primal program:

maximize
$$f \equiv \tfrac{1}{2}\left(\sum_{j=1} c_j n_j + \sum_{j=3}^{o} c_j' n_j'\right)$$

subject to
$$\sum_{j=1} n_j + \sum_{j=3}^{o} n_j' = n,$$
$$\sum_{j=1} (j-1)n_j + \sum_{j=3}^{o} (j-1)n_j' \leqslant k^2,$$
$$\sum_{j=3}^{o} n_j' \leqslant k,$$
$$n_j \geqslant 0,$$
$$n_j' \geqslant 0.$$

Dual program:

minimize
$$g \equiv nx + k^2 y + kz$$

subject to
$$x + (j-1)y \geqslant \tfrac{1}{2}c_j \quad (j = 1, 2, \ldots),$$
$$x + (j-1)y + z \geqslant \tfrac{1}{2}c_j' \quad (j = 3, 5, \ldots),$$
$$y \geqslant 0,$$
$$z \geqslant 0.$$

L_2 distance:

$$\hat{\mu}(\alpha)\begin{cases} = \min\left(\dfrac{1}{\sqrt{2}\alpha} + \dfrac{(\sqrt{5}-\sqrt{2})\alpha}{2}, \dfrac{3\sqrt{5}-\sqrt{13}}{4\alpha} + \dfrac{(\sqrt{13}-\sqrt{5})\alpha}{4}\right), & \alpha \leqslant \sqrt{2}, \\ \leqslant \min\left(\dfrac{3\sqrt{5}-\sqrt{13}}{4\alpha} + \dfrac{(\sqrt{13}-\sqrt{5})\alpha}{4}, \dfrac{2\sqrt{13}-3}{4\alpha} + \dfrac{\alpha}{4}\right), & \alpha > \sqrt{2}, \end{cases}$$

$$\hat{\mu}_0 \leqslant (\sqrt{65} - 7)^{1/2} \doteqdot 1.031.$$

For $\alpha \leqslant \sqrt{2}$, the bound is tight, since it is attained by a configuration with

$$\hat{n}_1 \approx (1-\alpha^2)n, \quad \hat{n}_2 \approx \alpha^2 n, \quad \hat{n}_j = 0 \quad (j \neq 1,2) \text{ for } \alpha \leqslant 1;$$
$$\hat{n}_2 \approx \tfrac{1}{2}(3-\alpha^2)n, \quad \hat{n}_4 \approx \tfrac{1}{2}(\alpha^2-1)n, \quad \hat{n}_j = 0 \quad (j \neq 2,4) \text{ for } 1 \leqslant \alpha \leqslant \sqrt{2}.$$

On the other hand, the bound is not tight for $\alpha > \sqrt{2}$. In fact, in the optimal solution of the primal program:

$$\hat{n}_2 \approx \tfrac{1}{2}(3-\alpha^2)n, \quad \hat{n}_4 \approx \tfrac{1}{2}(\alpha^2-1)n, \quad \hat{n}_j = 0 \quad (j \neq 2,4) \text{ for } \sqrt{2} < \alpha \leqslant \sqrt{3};$$
$$\hat{n}_4 \approx n - O(n^{1/2}), \quad \hat{n}_\infty \approx O(n^{1/2}), \quad \hat{n}_j = 0 \quad (j \neq 4,\infty) \text{ for } \alpha \geqslant \sqrt{3};$$

where $\infty = O(n^{1/2})$, the $\hat{n}_4$ edges should have L_2 length $\sqrt{13}/k$, which is easily seen to be impossible. The bound can be improved to $\hat{\mu}_0 \leqslant 1.014$ by modifying the linear program (see refined analysis below). A lower bound

$$\hat{\mu}_0 \geqslant \tfrac{1}{2}(2\sqrt{5}-1)^{1/2} \doteqdot 0.932$$

may be obtained by considering a configuration with

$$n_2 \approx [(j_o-\alpha^2-1)/(j_o-2)]n, \quad n_{j_o} \approx [(\alpha^2-1)/(j_o-2)]n, \quad n_j = 0 \quad (j \neq 2, j_o),$$

where j_o is a sufficiently large even integer.

L_∞ distance:

$$\hat{\mu}(\alpha) = \min\left(\frac{1}{2\alpha} + \frac{\alpha}{2}, \frac{3}{4\alpha} + \frac{\alpha}{4}\right), \quad \hat{\mu}_0 = \sqrt{3}/2 \doteqdot 0.866, \quad \hat{\alpha}_0 = \sqrt{3} \doteqdot 1.732.$$

The asymptotic supremum $\hat{\mu}(\alpha)$ in the L_∞ distance is attained, e.g., by a configuration with

$$\hat{n}_1 \approx (1-\alpha^2)n, \quad \hat{n}_2 \approx \alpha^2 n, \quad \hat{n}_j = 0 \quad (j \neq 1,2) \text{ for } \alpha \leqslant 1;$$
$$\hat{n}_2 \approx [(j_o-\alpha^2-1)/(j_o-2)]n, \quad \hat{n}_{j_o} \approx [(\alpha^2-1)/(j_o-2)]n,$$
$$\hat{n}_j = 0 \quad (j \neq 2, j_o) \text{ for } \alpha \geqslant 1,$$

where j_o is an even integer greater than $2\alpha^2$.

d. With/Without Preprocess and With Tour (Refinement)

To get a sharper bound for $\hat{\mu}(\alpha)$ in the L_2 distance, let us look into the cases of $j = 4, 6, 8, \ldots$, more closely. It is readily noted that there are two different configurations of j consecutive cells in the spiral-rack cell order, as illustrated in Figure 5 for $j = 6$. In correspondence with the two different configurations, we redefine two bounds c_j and

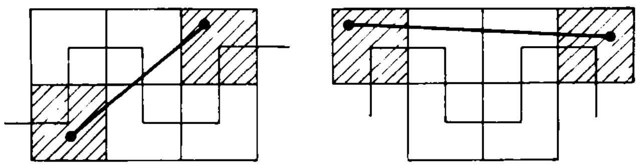


FIG. 5. Two equiprobable configurations for $j = 6$ in the spiral-rack cell order.

c_j' for $j = 4, 6, 8, \ldots$, as follows:

$$c_j = \begin{cases} [1 + (\frac{1}{2}j)^2]^{1/2}/k & (j = 4, 8, 12, \ldots), \\ [4 + (\frac{1}{2}j)^2]^{1/2}/k & (j = 6, 10, 14, \ldots); \end{cases}$$

$$c_j' = \begin{cases} \{4 + [\frac{1}{2}(j+2)]^2\}^{1/2}/k & (j = 4, 8, 12, \ldots), \\ \{1 + [\frac{1}{2}(j+2)]^2\}^{1/2}/k & (j = 6, 10, 14, \ldots). \end{cases}$$

Note that the c_j' defined here are the c_j defined at the beginning of this section. The modified linear program involves variables n_j $(j = 1, 2, 3, \ldots)$ and n_j' $(j = 3, 4, 5, \ldots)$ and contains one additional constraint, where Σ^e denotes summation over even integers.

Primal program:

maximize
$$f \equiv \tfrac{1}{2}\left(\sum_{j=1} c_j n_j + \sum_{j=3} c_j' n_j'\right)$$

subject to
$$\sum_{j=1} n_j + \sum_{j=3} n_j' = n,$$
$$\sum_{j=1} (j-1) n_j + \sum_{j=3} (j-1) n_j' \leqslant k^2,$$
$$\sum_{j=3}^{o} n_j' \leqslant k,$$
$$\sum_{j=4}^{e} j n_j' + \sum_{j=3}^{o} (j-1) n_j \leqslant k^2,$$
$$n_j \geqslant 0,$$
$$n_j' \geqslant 0.$$

Dual program:

minimize
$$g \equiv nx + k^2 y + kz + k^2 w$$

subject to

$$\begin{aligned} x &\geqslant \tfrac{1}{2}c_1, \\ x+(j-1)y &\geqslant \tfrac{1}{2}c_j \quad (j=2,4,6,\ldots), \\ x+(j-1)y+(j-1)w &\geqslant \tfrac{1}{2}c_j \quad (j=3,5,7,\ldots), \\ x+(j-1)y+z &\geqslant \tfrac{1}{2}c'_j \quad (j=3,5,7,\ldots), \\ x+(j-1)y+jw &\geqslant \tfrac{1}{2}c'_j \quad (j=4,6,8,\ldots), \\ y &\geqslant 0, \\ z &\geqslant 0, \\ w &\geqslant 0. \end{aligned}$$

L_2 distance:

$$\hat{\mu}(\alpha)\begin{cases} = \min\left(\dfrac{\sqrt{2}}{2\alpha}+\dfrac{(\sqrt{5}-\sqrt{2})\alpha}{2}, \dfrac{3\sqrt{5}-\sqrt{13}}{4\alpha}+\dfrac{(\sqrt{13}-\sqrt{5})\alpha}{4}\right), & \alpha\leqslant\sqrt{2}, \\ \leqslant \min\left(\dfrac{2\sqrt{5}-1}{4\alpha}+\dfrac{(\sqrt{13}+1-\sqrt{5})\alpha}{8}, \dfrac{2\sqrt{13}-3}{4\alpha}+\dfrac{\alpha}{4}\right), & \alpha>\sqrt{2}, \end{cases}$$

$$\hat{\mu}_0 \leqslant [\tfrac{1}{8}(2\sqrt{5}-1)(\sqrt{13}+1-\sqrt{5})]^{1/2} \doteqdot 1.014.$$

Thus the upper bound has been improved from 1.031 to 1.014, though it is still non-tight. (The corresponding α value is $\alpha = [2(2\sqrt{5}-1)/(\sqrt{13}+1-\sqrt{5})]^{1/2} \doteqdot 1.712$.) The optimal primal solution is

$$\begin{aligned} &\hat{n}_2 \approx (1-\tfrac{1}{4}\alpha^2)n-\hat{n}_{j_e}, \quad \hat{n}_{j_e} \approx (\tfrac{1}{2}\alpha^2-1)n/(j_e-2), \quad \hat{n}'_4 \approx \tfrac{1}{4}\alpha^2 n, \\ &\hat{n}_j = 0 \quad (j\neq 2, j_e), \qquad \hat{n}'_j = 0 \quad (j\neq 4) \text{ for } \sqrt{2}<\alpha\leqslant 2; \\ &\hat{n}'_4 \approx n-n_{j_e}-\hat{n}_{j_o}, \quad \hat{n}_{j_e} \approx [j_o(\alpha^2-4)+4]n/(j_oj_e-5j_o+4), \\ &\hat{n}_{j_o} \approx (j_e-\alpha^2-1)n/(j_oj_e-5j_o+4), \quad \hat{n}_j = 0 \quad (j\neq j_e, j_o), \\ &\hat{n}'_j = 0 \quad (j\neq 4) \text{ for } \alpha\geqslant 2, \end{aligned}$$

where j_e and j_o are, respectively, sufficiently large even and odd integers.

B. Bottom-Up Four-Square Algorithm

We shall confine our analysis to the case of the L_2 distance only. The case of the L_∞ distance is quite similar. Let n_j $(j=1,2,\ldots)$ be the number of points remaining unmatched at the beginning of the jth stage. At the very beginning of the algorithm, $n-n_1$ points are matched and the sum $\hat{M}_n^{(0)}$ of the distances of those pairs are bounded as

$$\hat{M}_n^{(0)} \leqslant \tfrac{1}{2}\sqrt{2}(n-n_1)/k.$$

At the jth stage where the cells are of size $2^j/k$, $n_j - n_{j+1}$ points are matched into pairs. The sum $\hat{M}_n^{(j)}$ of the distances of those pairs satisfies

$$\hat{M}_n^{(j)} \leq \tfrac{1}{2}\sqrt{2}\,2^j(n_j - n_{j+1})/k \qquad (j = 1, 2, \ldots).$$

Therefore we have

$$\hat{M}_n(k) = \hat{M}_n^{(0)} + \sum_{j=1} \hat{M}_n^{(j)}$$

$$\leq \frac{1}{\sqrt{2}k}\left(n + \sum_{j=1} 4^{j-1}(n_{2j-1} + 2n_{2j})\right). \tag{3}$$

Since each cell contains at most four points for $j = 1, 2, \ldots$, we have the inequality

$$n_{2j-1} + 2n_{2j} \leq 20k^2/4^{2j} \qquad (j = 1, 2, \ldots),$$

which is substituted into (3) to yield

$$\hat{M}_n(k) \leq \frac{1}{\sqrt{2}}\left(\frac{n}{k} + \frac{5k}{3}\right). \tag{4}$$

The right-hand side of (4) takes the minimum when k is a power of 2 lying between $\sqrt{\frac{3}{10}}n^{1/2}$ and $\sqrt{\frac{6}{5}}n^{1/2}$, so that we have

$$\hat{M}_n(k) \leq \tfrac{1}{2}\sqrt{15}\,n^{1/2} \doteqdot 1.936n^{1/2}.$$

On the other hand, the configuration of n points shown in Figure 6, which is constructed recursively as follows, will give the asymptotic lower bound $1.789n^{1/2}$ for $\hat{M}_n(k)$. Initially, we put four points in the corners of the unit square. The points in the diagonally opposite corners are to be matched. At a general step, we divide every cell

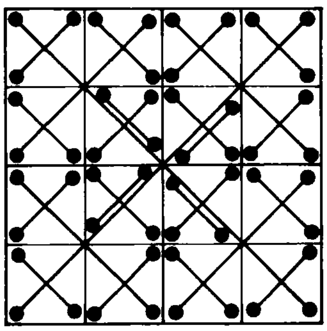


FIG. 6. Configuration which gives a lower bound on $\hat{\mu}$ for the bottom-up four-square algorithm.

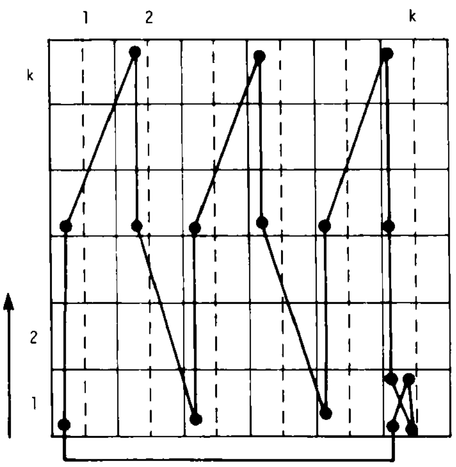


FIG. 7. Configuration which gives a lower bound on $\hat{\mu}$ for the strip-with-bucket algorithm.

into four cells, for each of which we put four or two points according as it contains no point or one point: (i) if it contains no point, we put four points in the four corners (the points in the diagonally opposite corners are to be matched); (ii) if it contains one point, we put two points in the diagonally opposite empty corners.

C. Strip-with-Bucket Algorithm

An upper bound for the serpentine algorithm with tour and without preprocess is also an upper bound for the strip-with-bucket algorithm. A lower bound for $\hat{M}_n^{(k)}$ is obtained by considering the configuration shown in Figure 7, where two points are distributed in each of the left half of the strips, one in the middle and the other at an end, the remaining $n - 2k$ points being laid in the left half of a cell in the corner. The horizontal shift has no effect on the resulting tours, which, in the asymptotic sense, may possibly contain $2k$ edges of L_2 or L_∞ length $\frac{1}{2}$ and $n - 2k$ edges of L_2 length $\sqrt{5}/2k$ (L_∞ length $1/k$). Thus we have

$$\hat{M}_n(k) \geqslant (\sqrt{5}/2k)(\tfrac{1}{2}n - k) + \tfrac{1}{2}k \geqslant 1.057n^{1/2} - 1/\sqrt{2}$$

for the L_2-distance, and

$$\hat{M}_n(k) \geqslant (1/k)(\tfrac{1}{2}n - k) + \tfrac{1}{2}k \geqslant n^{1/2} - 1$$

for the L_∞-distance.

IV. AVERAGE-CASE ANALYSIS

Throughout this section, we shall assume that n points are randomly distributed uniformly in a unit square. Let us denote by M_n the random variable representing the cost of a matching, by $\mu = \mu(\alpha)$ the asymptotic expectation of $M_n/n^{1/2}$, and by $\mu_0 = \mu(\alpha_0) = \min_\alpha \mu(\alpha)$ the optimal value of μ.

A. Theoretical Estimates

It is known [7, 9] that the minimum cost in the Euclidean distance of the perfect matchings of n points, when divided by $n^{1/2}$, converges in probability to a constant which lies between 0.25 and 0.401. We shall evaluate the expected costs of the approximate solutions obtained by means of the algorithms proposed in Sec. II.

Elementary probability calculation will yield the following relations. The probability for a cell to contain j points is

$$p_j = \binom{n}{j}\left(\frac{1}{k^2}\right)^j\left(1 - \frac{1}{k^2}\right)^{n-j},$$

or, as $n \to \infty$ with $k = \alpha n^{1/2}$,

$$p_j \sim \frac{1}{j!}\,\frac{1}{\alpha^{2j}}\exp(-1/\alpha^2) \qquad (j = 0, 1, \ldots).$$

Thus, the expected number m_p of cells containing at least one point is

$$m_p \sim k^2[1 - \exp(-1/\alpha^2)],$$

and the expected number m_o of cells containing an odd number of points is

$$m_o \sim \tfrac{1}{2}k^2[1 - \exp(-2/\alpha^2)].$$

The expected cost of a straightforward algorithm without preprocess is asymptotically

$$E[M_n] \sim \frac{n - m_p}{2}\tilde{c}_1 + \frac{m_p}{2}\sum_{j=2}\frac{m_p}{k^2}\left(1 - \frac{m_p}{k^2}\right)^{j-2}\tilde{c}_j, \tag{5}$$

where the $\tilde{c}_j$ are the expected distances between two points contained in cells at cell distance $j - 1$ (see Appendix). The straightforward algorithms with preprocess can be treated quite similarly; the above expression (5) remains valid if m_p is replaced by m_o.

The results of our Monte Carlo experiment suggest that the cost of an n-point matching constructed by an approximation algorithm, when divided by $n^{1/2}$, will converge in probability to a constant as n tends to infinity, just as the exact minimum cost does. If that is the case, employing a tour will lead us to no substantial improvement on the resulting cost when n is sufficiently large, since, for n large enough, the cost of the two matchings contained in a tour cannot be significantly different. The optimal values α_0 of α minimizing the expected costs are calculated on the basis of the above formula (5) and are given in Table I. The theoretical estimates of μ_0 and α_0 for the bottom-up four-square and the strip-with-bucket algorithms are also given.

The spiral-rack algorithm with preprocess (with or without tour) seems to be the best in average performance among the linear-time algorithms proposed in the present article. Taking into account also the worst-case behavior, it may be recommended to use, for the actual application to the plotter problem, the spiral-rack algorithm with preprocess and with tour, where we may set $\alpha = 1.29$ for the L_2 distance problem and $\alpha = 1.26$ for the L_∞ distance problem. Then we have $\mu = 0.490$ and $\hat{\mu} \leqslant 1.05$ for the L_2 distance and $\mu = 0.449$ and $\hat{\mu} \leqslant 0.91$ for the L_∞ distance, these μ and $\hat{\mu}$ both remaining within a few percent increase compared with the optimal values. The tour requires more computation, but, for n small, it yields a better solution.

B. Experimental Results

The algorithms have been implemented on the HITAC M-200H system at the Computer Centre of the University of Tokyo. Data are given in two real arrays of size n in terms of the coordinates of n points. The program (subroutine) returns the matching in an integer array of size n, the jth element of which represents the number of its partner point. A straightforward algorithm uses for the working area only an integer array of size $k \times k$ as well as several simple variables.

In the Monte Carlo experiment with 200 samples for each n, it is observed that (i) the expectation of the normalized cost $M_n/n^{1/2}$ tends to the theoretical estimate and the variance of $M_n/n^{1/2}$ diminishes as $O(1/n)$ (Fig. 8), (ii) the algorithms with tour require 1.1 ~ 2.6 times as much computing time as the algorithms without tour do, and (iii) the strip algorithm does not run in linear time even on the average (Fig. 9). The recommended algorithm seems superior in the light of the tradeoff between the (average- and worst-case) cost and the computing time (Figs. 10, 11). In Figure 12 the ratio of the cost of the approximate solution by the recommended algorithm to that of the exact solution is shown for $n = 16, 32, 64, 128, 256$ (20 samples for each n), where it is observed that the former is about 1.5 times larger than the latter.

V. APPLICATION TO THE DRAWING OF A ROAD MAP BY A MECHANICAL PLOTTER

When drawing a connected graph by a mechanical plotter, the wasted plotter-pen movement is minimized by finding a minimum-weight perfect matching of the vertices

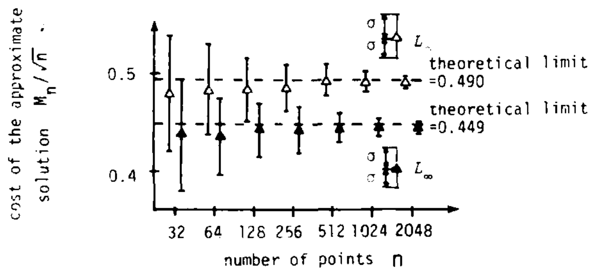


FIG. 8. Expected values and standard deviations of the costs of the matchings constructed by the recommended approximation algorithm.

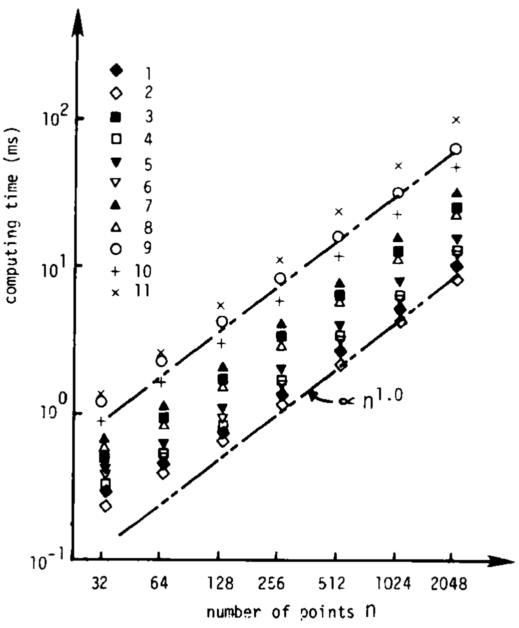


FIG. 9. Computing time of the approximation algorithms (L_2 distance) (cf. Table I).

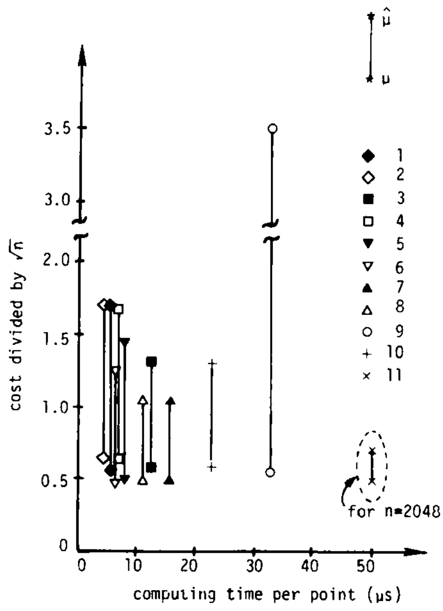


FIG. 10. Computing time and average- and worst-case costs of approximate solutions (L_2 distance) (cf. Table I).

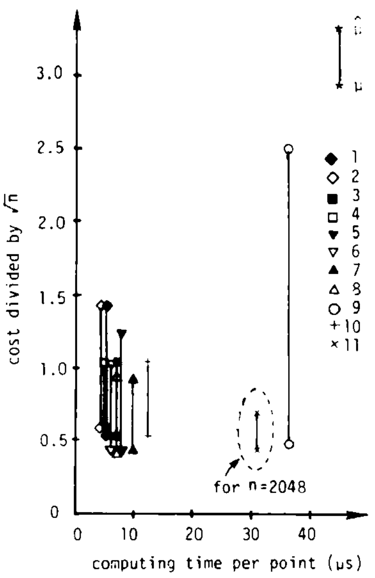


FIG. 11. Computing time and average- and the worst-case costs of approximate solutions (L_∞ distance) (cf. Table I).

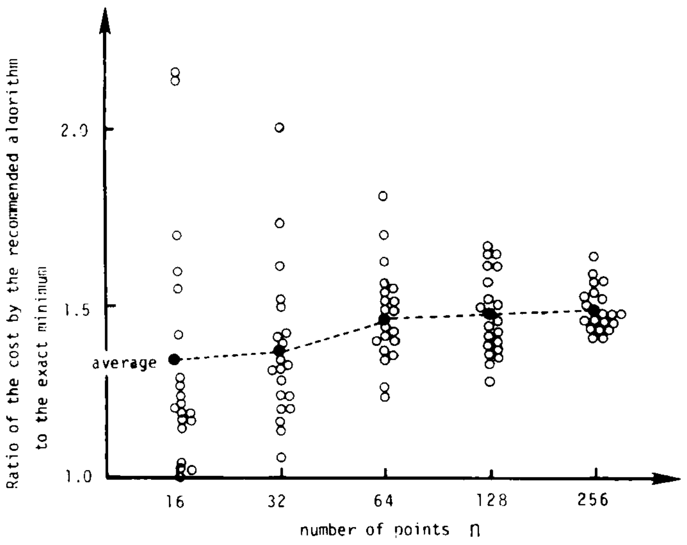
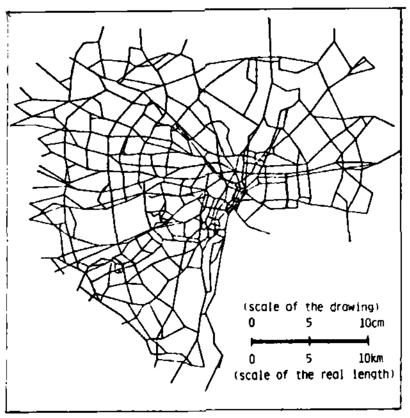
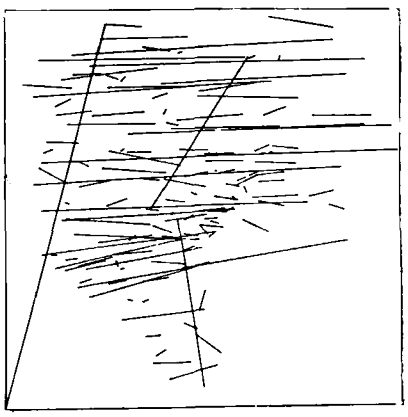


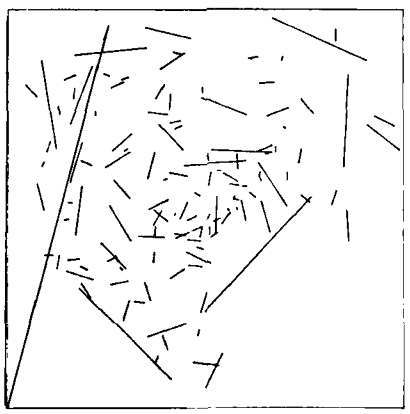
FIG. 12. Ratio of the cost of the approximate solution by the recommended algorithm to the exact minimum (20 samples for each n; L_2 distance).



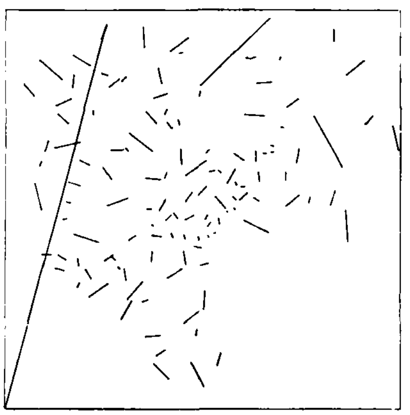
(i) Map drawn



(ii) Wasted pen movement when odd-degree vertices are paired as they are input



(iii) Wasted pen movement by the spiral rack algorithm with preprocess and with tour (L_∞-distance)



(iv) Wasted pen movment by the exact algorithm (L_∞-distance)

FIG. 13. Drawing the road map of the Tokyo city area.

of odd degrees, adding them to the original graph, and traversing the Eulerian path on the extended graph, where the added edges are to be traversed with the pen off the paper [2, 3, 4, 5, 8]. We applied this technique to drawing the road map of the Tokyo city area [Fig. 13(i)]. The computing time and the lengths of pen movement are shown in Table II. The wasted pen movement is depicted in Figures 13(ii)-13(iv) for some of the different plotting strategies when the L_∞ distance is adopted as the distance measure. In [3], unconnected graphs are also considered and an experiment on a graph with about 2×10^4 vertices and as many edges is shown.

TABLE II. An application: Drawing the road map of the Tokyo city area [5], 850 edges, 513 vertices (including 254 vertices of odd degree).

Plotting Strategy		CPU Time (ms): For Making a Matching	CPU Time (ms): For Finding an Eulerian Path	CPU Time (ms): Total	Pen Movement (cm): Wasted with Pen Off (μ^b)	Pen Movement (cm): 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 algorithm (tour) No. 3	L_2	3.1	15	18	238 (0.43)	896
	L_∞	2.3	15	18	206 (0.37)	817
Spiral-rack algorithm (preprocess, tour) No. 8	L_2	3.0	15	18	227 (0.41)	896
	L_∞	1.9	15	17	214 (0.39)	817
Strip algorithm No. 11	L_2	12	15	27	183 (0.33)	896
	L_∞	9	15	24	164 (0.30)	817
Exact algorithm[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](Total length of the wasted pen movement scaled to a unit square)/$n^{1/2}$.

VI. CONCLUSION

We proposed linear-time approximation algorithms for finding the minimum-weight perfect matching on a plane and investigated their performances theoretically and experimentally. The spiral-rack algorithm with preprocess and with tour is recommended for practical purposes. It was seen that a considerable improvement in the efficiency of drawing graphs by a mechanical plotter is realized by using the recommended algorithm in combination with a linear-time (exact) algorithm for Euler's problem.

The authors thank Professor Z. Galil of Tel-Aviv University and Professor D. Avis of McGill University for valuable information concerning the subject and for stimulating discussions. They are indebted to Dr. A. Taguchi of Yamanashi University for the road-map data as well as for valuable suggestions. Thanks are due also to Mr. H. Imai for a program of the exact algorithm for the minimum-weight matching and for collaboration in calculating the expected performance.

APPENDIX. CALCULATION OF EXPECTED LENGTHS $\tilde{c}_j$

The expected length $\tilde{c}_j$ of an edge connecting two points in cells at cell distance $j - 1$ in the serpentine and the spiral-rack cell order may be calculated by elementary methods in probability (see [4] for details).

To begin with, we calculate the expected distance $D(x, y)$ between two points distributed uniformly in a rectangle of sides x and y to get

L_2 distance:

$$D(x,y) = \frac{x^5 + y^5 - (x^4 - 3x^2y^2 + y^4)(x^2 + y^2)^{1/2}}{15x^2y^2} + \frac{xy^4 \log\{[x + (x^2 + y^2)^{1/2}]/y\} + x^4 y \log\{[y + (x^2 + y^2)^{1/2}]/x\}}{6x^2y^2},$$

L_∞ distance: $$D(x,y) = \frac{x}{3} + \frac{y^2}{6x} - \frac{y^3}{30x^2} \qquad (x \geqslant y).$$

In particular, we put

$$D(x,0) = \lim_{y \to 0} D(x,y) = \tfrac{1}{3}x \qquad \text{(in either distance).}$$

Then we define for $j \geqslant 2$

$$d_j = \tfrac{1}{2}j^2 D(1,j) - (j-1)^2 D(1,j-1) + \tfrac{1}{2}(j-2)^2 D(1,j-2).$$

Serpentine Cell Order

The expected length $\tilde{c}_j$ in the serpentine cell order is given by

$$\tilde{c}_1 = D(1,1)/k, \qquad \tilde{c}_j = d_j/k \qquad (j = 2, 3, \ldots),$$

which may be simplified, in the case of the L_∞ distance, to

$$\tilde{c}_1 = 7/15k, \qquad \tilde{c}_2 = 61/60k, \qquad \tilde{c}_j = (j-1)/k \qquad (j = 3, 4, \ldots).$$

Spiral-Rack Cell Order

Obviously we have

$$\tilde{c}_1 = D(1,1)/k, \qquad \tilde{c}_2 = d_2/k.$$

For even $j \geqslant 4$, two types of the configuration, as shown in Figure 5 for $j = 6$, are equiprobable and the desired expectation $\tilde{c}_j$ is the average of the expected distances corresponding to the two configurations. Specifically, we have for $j = 1, 2, \ldots,$

$$\tilde{c}_{4j-1} = e_{2j}/k,$$
$$\tilde{c}_{4j} = (d_{2j} + e_{2j+1})/2k,$$
$$\tilde{c}_{4j+1} = d_{2j+1}/k,$$
$$\tilde{c}_{4j+2} = (e_{2j+1} + d_{2j+2})/2k,$$

where e_j is defined as

$$e_j = j^2 D(2,j) - 2(j-1)^2 D(2,j-1) + (j-2)^2 D(2,j-2) - d_j.$$

For the L_∞ distance, the above expressions are reduced to

$$\tilde{c}_1 = 7/15k, \qquad \tilde{c}_2 = 61/60k, \qquad \tilde{c}_3 = 37/30k, \qquad \tilde{c}_4 = 121/80k,$$
$$\tilde{c}_5 = 2/k, \qquad \tilde{c}_6 = 601/240k, \qquad \tilde{c}_j = (j-1)/2k \qquad (j = 7, 8, \ldots).$$

References

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, Ma. (1974).

[2] 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.

[3] M. Iri, K. Murota, and S. Matsui, An approximate solution for the problem of optimizing the plotter pen movement. In *Proc. 10th IFIP Conference on System Modeling and Optimization Lecture Notes in Control and Information Sciences*, 38, A. V. Balakrishnan and M. Thoma, Eds. Springer-Verlag, New York (1982) 572–580.

[4] 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. Research Memorandum RMI 81-07. Department of Mathematical Engineering and Instrumentation Physics, University of Tokyo (1981).

[5] M. Iri and A. Taguchi, The determination of the pen-movement of an *XY*-plotter and its computational complexity (in Japanese). In *Proc. Spring Conference of the Operations Research Society of Japan, P-8*, 1980, pp. 204, 205.

[6] E. L. Lawler, *Combinatorial Optimization–Networks and Matroids.* Holt, Rinehart and Winston, New York (1976).

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

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

[9] J. M. Steele, Subadditive Euclidean functionals and non-linear growth in geometric probability. *Ann. of Probability* 9 (1981) 365–376.

[10] K. J. Supowit, D. A. Plaisted, and E. M. Reingold, Heuristics for weighted perfect matching. In *Proc. 12th Ann. ACM Symp. on Theory of Computing*, 1980, pp. 398–414.

Received February 26, 1981
Accepted May 11, 1982