Journal of the Operations Research
Society of Japan
Vol. 29, No. 1, March 1986

WORST-CASE ANALYSIS FOR PLANAR MATCHING AND TOUR HEURISTICS WITH BUCKETING TECHNIQUES AND SPACEFILLING CURVES

Hiroshi Imai
University of Tokyo

(Received March 13, 1985: Revised January 16, 1986)

Abstract The worst-case performance of heuristics with bucketing techniques and/or spacefilling curves for the planar matching problem and the planar traveling salesman problem is analyzed. Two types of heuristics are investigated, one is to sequence given points in a spacefilling-curve order and the other is to sequence the points in the order of buckets which are arranged according to the spacefilling curve. The former heuristics take $O(n \log n)$ time, while the latter ones run in $O(n)$ time when the number of buckets is $O(n)$. It is shown that the worst-case performance of the former and that of the latter are the same if a sufficient number of $O(n)$ buckets are provided, which is investigated in detail especially for the heuristics based on the Sierpiński curve. The worst-case performance of the heuristic employing the Hilbert curve is also analyzed.

1. Introduction

The planar matching problem is to find a minimum-length perfect matching of n (even) points in the plane, that is, to determine how to match the n points in pairs so as to minimize the sum of the lengths between the matched points. The planar traveling salesman problem, or simply the planar tour problem, is to find a tour (circuit) of minimum total length that visits each of n given points in the plane. Both problems have many applications in various fields. The planar tour problem is NP-complete (Papadimitriou [11]), while the planar matching problem can be solved in $O(n^3)$ time (Lawler [9]); however, from the practical point of view, even such an $O(n^3)$-time algorithm seems to be too complicated and take too much time for large-scale problems.

In order to solve the large-scale problems, fast and simple heuristics have been proposed. Concerning the matching heuristics, see a survey by Avis [2]. Iri, Murota and Matsui [6], [7], [8] have proposed linear-time heuristics for the planar matching problem in connection with the application of the problem to drawing a figure by a mechanical

plotter. Their algorithms use the bucketing techniques, run quite fast in practice and give satisfactory solutions (concerning the use of bucketing techniques in various problems, see [1]). The worst-case performance of these heuristics, which is analyzed in [8], is comparable to that of heuristics taking more time.

Bartholdi and Platzman [3], [4], [12] have posed an $O(n \log n)$-time algorithm for the planar traveling salesman problem, which uses spacefilling curves, especially, the Sierpiński curve. A spacefilling curve is a continuous mapping from the unit interval onto the unit square. The spacefilling curve itself forms a path, and their algorithm is to sequence the given points as they appear along the spacefilling curve and form a tour of the points in that order. The algorithm using the Sierpiński curve can be implemented in $O(n \log n)$ time, where the main part is sorting the given set of points along the curve. The worst-case performance for the Sierpiński-curve algorithm is theoretically analyzed in [12].

In this paper, we shall investigate the worst-case performance of heuristics with bucketing techniques and/or spacefilling curves for the planar matching and tour problems. Iri (e.g., see [1], [5]) pointed out that, given a spacefilling curve such as the Sierpiński curve and the Hilbert curve, we can devise linear-time heuristics using buckets the number of which is proportional to the number of points and which are arranged in the spacefilling curve order. When the Sierpiński curve is concerned, triangular buckets are introduced, which themselves are of theoretical interest. These heuristics are superior to those sequencing, directly, given points in the spacefilling curve order with respect to the time complexity; however, the worst-case performance of the former might become worse than that of the latter. We show that, if buckets are sufficiently provided, the worst-case performance of the bucketing heuristics is the same with that of spacefilling-curve heuristics. Especially, we give tight bounds for the bucketing heuristics in which buckets are sequenced in the Sierpiński curve order, where we adopt an approach similar to that used in [8] in order to analyze the worst-case performance. However, unlike in [8], it is not easy to obtain tight bounds for distances of any two points in two buckets, which is investigated in detail in the paper. The worse-case performance of the heuristic sequencing points in the Hilbert curve order is also analyzed, which has not yet been studied. In this analysis, we partly take an approach given in [8] and use a recurrence relation in the main part. Although we shall not go into details here for analyzing the worst-case performance of the bucket heuristics using the Hilbert curve, we can readily obtain bounds, which may be a little loose, for those heuristics by combining the above-mentioned results for the Hilbert-curve algorithm with the lemma given in section 2.

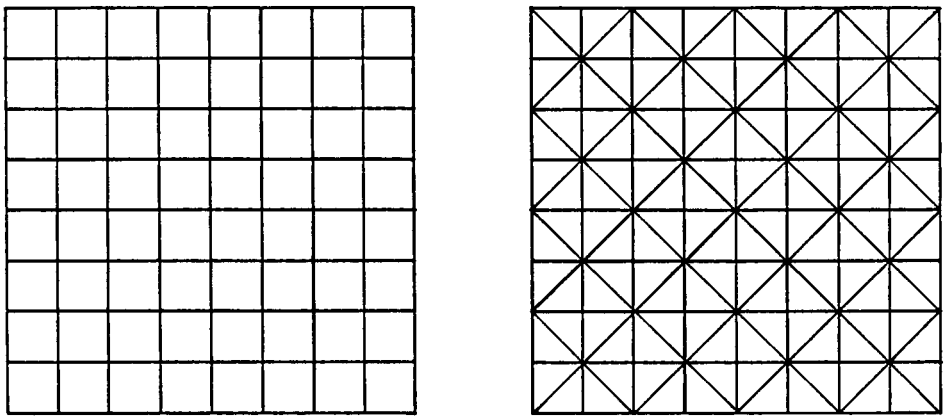
These analyses show that, with respect to the worst-case performance as defined in section 2, the spacefilling-curve heuristics and the bucket heuristics where buckets are ordered in the spacefilling-curve order are a little worse than the spiral-rack algorithm recommended in [8].

2. Heuristics for Planar Matchings and Tours

As fast heuristics for planar matchings and tours, the following two types of algorithms are known, where the first one runs in $O(n)$ time, and the second one runs in $O(n \log n)$ time for n points.

A. Bucket algorithms

These algorithms are proposed by Iri, Murota and Matsui [6], [7], [8]. In these algorithms, we partition the unit square, where n points are distributed, into subsquares or subtriangles, called *buckets*, as depicted in Fig.2.1. Each point belongs to one of those buckets. In the case the unit square is divided into $k \times k$ square buckets, we can determine the bucket to which a point belongs by multiplying the coordinates of the point by k and then truncating off the fractional parts, which can be done in a constant time. In the case of triangular buckets, this can be similarly determined in a constant time although it takes a little more time.

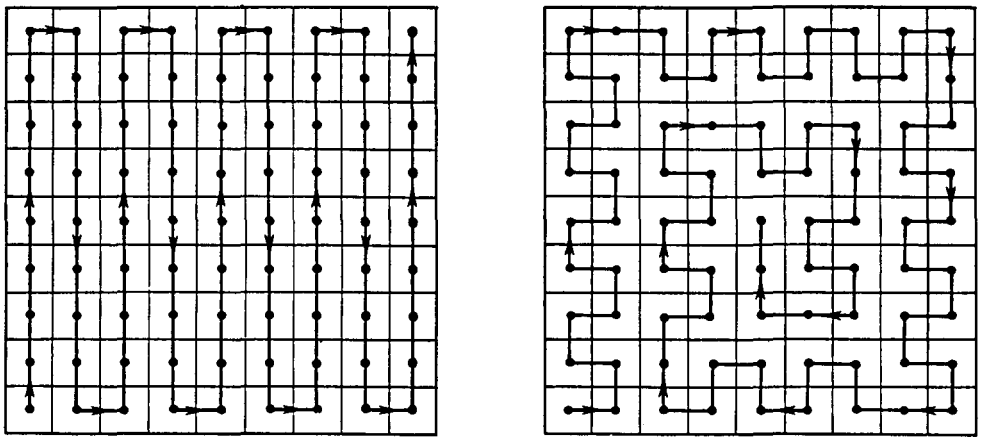


(a) square buckets (b) triangular buckets

Fig.2.1. Partition of the square into buckets

The buckets are ordered in a prescribed order (in Fig.2.2, two orders of square buckets, the serpentine order and the spiral-rack order, proposed in [8] are depicted). We number the n points so as to form a sequence which is consistent with the order of buckets the point belongs to; that is, points in the same bucket may arbitrarily be ordered among themselves, but points in different buckets must be ordered consistently with the order of the buckets. Then, for an approximate solution for matchings, we adopt the matching consisting of pairs of the $(2i-1)$st point and the $2i$th $(i = 1, 2, \ldots, n/2)$. For an approximate solution for tours, we simply connect the points in the above sequence where the first and the last points in the sequence is connected to form a tour.

Concerning the algorithm for matchings, the following variants of the algorithm are proposed.



(a) Serpentine order (b) Spiral-rack order

Fig.2.2. Two orders of square buckets [8]

(i) *Preprocessing*: Before ordering points, match a pair of points in the same bucket as much as possible (hence, in ordering the remaining points, each bucket contains at most one point).

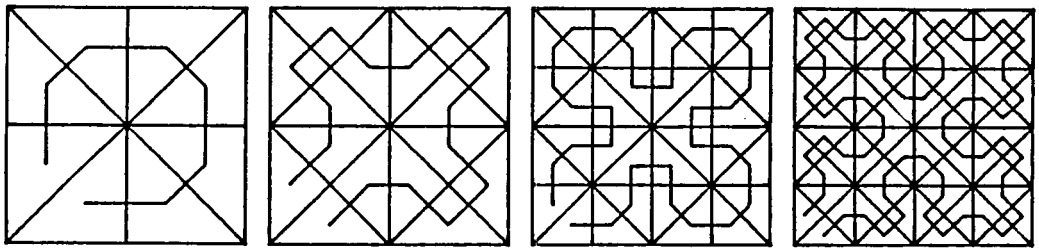
(ii) *Tour*: As an approximate solution, take the cheaper of the two, one is a matching consisting of pairs of the $(2i-1)$st and the $2i$th points, and the other is a matching consisting of pairs of the $2i$th and $(2i+1)$st points.

When the unit square is divided into $O(n)$ buckets and the order of buckets can be computed in $O(n)$ time, these algorithms run in $O(n)$ time, quite efficiently.

B. Spacefilling-curve algorithms

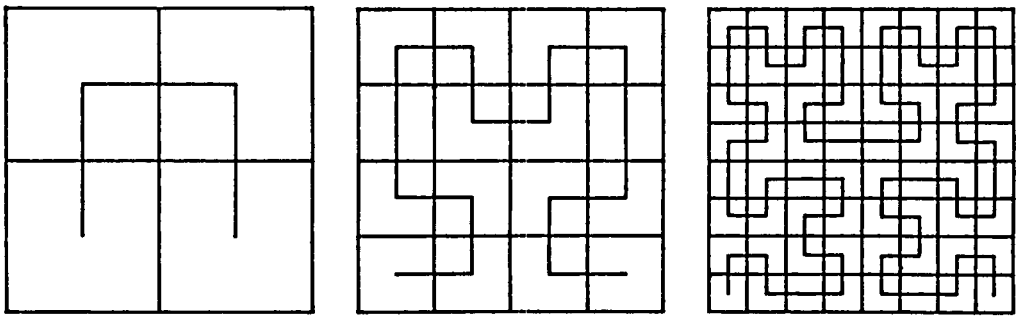
These kinds of algorithms are proposed by Bartholdi and Platzman [3], [4], [12]. In order to obtain an approximate tour, these algorithms sequence n points as they appear along a spacefilling curve, such as the Sierpiński curve and the Hilbert curve. As for matchings, simply match the $(2i-1)$st and the $2i$th points in the sequence; we can also apply the technique, *tour*, as described above to obtain a better matching. Since sorting the n points along the curve is needed, these algorithms take $O(n \log n)$ time.

Let us now describe algorithms which will be investigated in the paper. A spacefilling curve is a continuous mapping from the unit interval onto the unit square. As the spacefilling curve, the Sierpiński curve and the Hilbert curve are well known. The Sierpiński curves S_i of order i $(i = 3, 4, 5, 6)$ are defined as in Fig.2.3. S_∞ is a spacefilling curve, called the *Sierpiński curve*. The Hilbert curves H_i of order i $(i = 1, 2, 3)$ are defined as in Fig.2.4, and H_∞ is a spacefilling curve, referred to as the *Hilbert curve*. The worst-case performance of the spacefilling-curve algorithm with the Sierpiński curve, to be called a *Sierpiński-curve algorithm*, is almost analyzed by Platzman and Bartholdi [12], while that with the Hilbert curve, to be called a *Hilbert-curve algorithm*, has not yet been studied. In



(a) S_3 (b) S_4 (c) S_5 (d) S_6

Fig.2.3. Sierpiński curves S_i of order i ($i = 3, 4, 5, 6$)



(a) H_1 (b) H_2 (c) H_3

Fig.2.4. Hilbert curves H_i of order i ($i = 1, 2, 3$)

section 4, we shall investigate the worst-case performance of the Hilbert-curve algorithm.

In Fig.2.3, we draw the partition of the unit square into congruent isosceles right triangles, called ***triangular buckets***, with the curves themselves so that the definitions of the curves are easy to understand. As is seen from the figure, an order of triangular buckets is naturally introduced; that is, buckets are ordered as they are traversed by the Sierpiński curve of order i, which is referred to as a *Sierpiński bucket order* of triangular buckets. Then, we can consider the bucket algorithm with the Sierpiński bucket order as in the way described above, which is referred to as a *Sierpiński-bucket algorithm*. In this algorithm, we partition the unit square into $2^{\lceil \log_2 \alpha^2 n \rceil}$ triangular buckets with a parameter α. The order of buckets can be computed in linear time by the "folding-over" algorithm, similar to one in [10]. This algorithm with taking α infinitely large coincides with the Sierpiński-curve algorithm. As α grows larger, the worst-case performance of the Sierpiński-bucket algorithm would become better, but it comes to take more time and space. In section 3, we shall investigate the worst-case performance of the Sierpiński-bucket algorithm in detail.

The worst-case performance of a heuristic is estimated not by the worst-case ratio of a solution obtained by the heuristic and an optimum solution, but by the worst-case absolute value of a solution obtained by the heuristic for all possible configurations of n points in the unit square. For a fixed algorithm for matchings, let $\widehat{M}_n$ be the supremum of the costs

of matchings obtained by the algorithm over all possible configurations of n points in the unit square, and put

$$\hat{\mu}_O = \limsup_{n\to\infty} \widehat{M}_n/\sqrt{n}. \tag{2.1}$$

The efficiency of algorithms using buckets depends heavily upon the number of buckets employed in the algorithms, so that, for an algorithm using $\alpha^2 n$ buckets, let $\widehat{M}_n(\alpha)$ be the supremum of the costs of matchings obtained over all possible configurations of n points in the unit square, and put

$$\hat{\mu}_O(\alpha) = \limsup_{n\to\infty} \widehat{M}_n(\alpha)/\sqrt{n}, \qquad \hat{\mu}_O = \hat{\mu}_O(\hat{\alpha}_O) = \min_{\alpha} \hat{\mu}_O(\alpha). \tag{2.2}$$

$\hat{\mu}_O$ can be considered to be the efficiency of the algorithm in the worst case.

For a fixed algorithm for tours, let $\widehat{T}_n$ be the supremum of the costs of tours obtained by the algorithm over all possible configurations of n points in the unit square, and further, define $\widehat{T}_n(\alpha)$, $\hat{\tau}_O$ and $\hat{\tau}_O(\alpha)$ similarly as $\widehat{M}_n(\alpha)$, $\hat{\mu}_O$ and $\hat{\mu}_O(\alpha)$, respectively.

Concerning the distance, we consider not only the L_2 distance but also the L_∞ distance, since the L_∞ distance is suitable in applying the matching heuristic to the problem of drawing a figure by a mechanical plotter efficiently [7], [8]. In Table 2.1, we summarize the results obtained in this paper with some of the previously known results.

In concluding this section, we consider the relation of the worst-case performance of a spacefilling-curve heuristic and that of a bucket heuristic employing the same spacefilling-curve. The upper bounds, which may loose, of $\hat{\mu}_O(\alpha)$ and $\hat{\tau}_O(\alpha)$ of the bucket heuristic can be obtained from $\hat{\mu}_O$ and $\hat{\tau}_O$ of the spacefilling-curve heuristic, which we shall show in the following lemma.

Lemma 2.1. Let μ and τ be $\hat{\mu}_O$ and $\hat{\tau}_O$, respectively, of the spacefilling-curve heuristic H_T. Consider the bucket heuristic H_B where $\alpha^2 n$ buckets are ordered by the spacefilling curve. Let $\frac{c}{\alpha\sqrt{n}}$ be the maximum distance of two points in the same bucket in H_B, where c is a constant. Then, $\hat{\mu}_O(\alpha)$ and $\hat{\tau}_O(\alpha)$ of H_B are bounded as follows.

(i) Concerning $\hat{\tau}_O(\alpha)$, we have

$$\hat{\tau}_O(\alpha) \le \begin{cases} \dfrac{\tau}{\sqrt{2}} + \dfrac{c}{\alpha} & (0 < \alpha \le \dfrac{2\sqrt{2}c}{\tau}) \\ \dfrac{\alpha\tau^2}{8c} + \dfrac{2c}{\alpha} & (\dfrac{2\sqrt{2}c}{\tau} \le \alpha \le \dfrac{4c}{\tau}) \\ \tau & (\alpha \ge \dfrac{4c}{\tau}) \end{cases} \tag{2.3}$$

(Note that for the planar matching heuristic without preprocess and with tour, $\hat{\mu}_O(\alpha) \le \hat{\tau}_O(\alpha)/2$).

(ii) For the matching heuristic with preprocess and without tour, we have

$$\hat{\mu}_O(\alpha) = \begin{cases} \dfrac{\alpha\mu^2}{2c} + \dfrac{c}{2\alpha} & (0 < \alpha \le \dfrac{c}{\mu}) \\ \mu & (\alpha \ge \dfrac{c}{\mu}) \end{cases} \tag{2.4}$$

Table 2.1. Asymptotic supremum $\hat{\mu}_O$, for the optimum value $\hat{\alpha}_O$, and $\hat{\mu}_O(\alpha)$ of (the costs of matchings)/$\sqrt{n}$

Algorithm	L_2 distance		L_∞ distance	
	$\hat{\mu}_O$	$\hat{\alpha}_O$	$\hat{\mu}_O$	$\hat{\alpha}_O$
Sierpiński-curve[a]	1.414	–	1.414	–
Sierpiński-curve[a] with tour	1[c]	–	1	–
Hilbert-curve[a] with tour	≤ 1.105 ≥ 1.045	–	≤ 1.011 ≥ 1.006	–
Spiral-rack[b,d] with preprocess, tour	≤ 1.014 ≥ 0.932	[1.712][e]	0.866	1.732
	$\hat{\mu}_O(\alpha)$	α	$\hat{\mu}_O(\alpha)$	α
Sierpiński-bucket[b,f] with preprocess	$\frac{\alpha}{2}+\frac{1}{\alpha}$	$\alpha \leq 2$	$\sqrt{2}(\frac{\alpha}{2}+\frac{1}{2\alpha})$	$\alpha \leq 1$
	$\sqrt{2}$	$\alpha \geq \sqrt{2}$	$\sqrt{2}$	$\alpha \geq 1$
Sierpiński-bucket[b,f] with tour	$\leq \sqrt{1+1/\alpha^2}$	$\alpha \leq 2\sqrt{2}$	$\sqrt{2}(\frac{\alpha}{4}+\frac{1}{2\alpha})$	$\alpha \leq \sqrt{2}$
	$\leq \frac{\alpha}{8}+\frac{2}{\alpha}$	$2\sqrt{2} \leq \alpha \leq 4$	1	$\alpha \geq \sqrt{2}$
	1	$\alpha \geq 4$		

[a]$O(n \log n)$-time algorithm, [b]$O(n)$-time algorithm, [c]From [4], [d]From [8], [e]The value corresponding to the upper bound of $\hat{\mu}_O$, [f]The number of buckets is $2k^2$ where $k = 2^{\lceil \log_2 \alpha\sqrt{n} \rceil - 1/2}$.

(iii) For the matching heuristic without preprocess and without tour, we have

$$\hat{\mu}_O(\alpha) \leq \mu + \frac{c}{\alpha} . \tag{2.5}$$

Proof: (i) For sufficiently many n points distributed in the unit square, let n_b be the number of buckets containing at least two points, and n' be the number of points contained in such buckets, where $n' \geq 2n_b$. For each bucket b_i containing at least two points, let $P_{i,1}$, $P_{i,2}$ be the first and the last points in b_i that appears in the obtained tour T, and $P_{i,3}$ be the next point of $P_{i,2}$ in the tour. Update the edge set of T by replacing edge $\{P_{i,2}, P_{i,3}\}$ by edges $\{P_{i,1}, P_{i,2}\}$ and $\{P_{i,1}, P_{i,3}\}$, and denote the resultant edge set by T'. By the triangle inequality, the total length of the tour T is bounded by that of T'. The total length of T' is bounded by (the maximum possible length of the tour of $n - n' + n_b$ points in the unit square formed by the spacefilling-curve heuristic)$+n' \times$(the maximum

possible distance of two points in the same bucket). Hence, for the heuristic H_B, we have

$$\begin{aligned}\widehat{T}_n(\alpha) &\le \tau\sqrt{n-n'+n_b}+\frac{cn'}{\alpha\sqrt{n}} \le \tau\sqrt{n-n'/2}+\frac{cn'}{\alpha\sqrt{n}}\\ &= \sqrt{n}(\tau\sqrt{1-x/2}+\frac{cx}{\alpha}),\end{aligned}$$

where $x = n'/n$, $0 \le x \le 1$. Hence,

$$\widehat{\tau}_O(\alpha) \le \max_{0\le x\le 1}\{\tau\sqrt{1-x/2}+\frac{cx}{\alpha}\}.$$

Solving this, we obtain (2.3).

(ii) Let n' be the number of points matched in preprocessing. Then, we have

$$\widehat{M}_n(\alpha) \le \mu\sqrt{n-n'}+\frac{n'}{2}\frac{c}{\alpha\sqrt{n}} = \sqrt{n}(\mu\sqrt{1-x}+\frac{cx}{2\alpha}), \tag{2.6}$$

where $x = n'/n$. Hence, $\widehat{\mu}_O(\alpha) \le \max_{0\le x\le 1}\{\mu\sqrt{1-x}+\frac{cx}{2\alpha}\}$, and, solving this, we obtain (2.4). As is seen from (2.6), this bound is tight.

(iii) Let n' be the number of points each of which is matched with a point in the same bucket, and let n_b be the number of buckets containing two points which are matched with points in the other buckets ($n' + 2n_b \le n$). Then, we have

$$\widehat{M}_n(\alpha) \le \frac{n'}{2}\frac{c}{\alpha\sqrt{n}}+\mu\sqrt{n-n'}+2n_b\frac{c}{\alpha\sqrt{n}} \le (\mu+\frac{c}{\alpha})\sqrt{n},$$

and hence $\widehat{\mu}_O(\alpha) \le \mu + \frac{c}{\alpha}$. □

From this lemma, it is seen that, except the matching heuristic without preprocessing and without tour, the worst-case performance of the spacefilling-curve heuristics and that of the bucket heuristics are the same if α is taken to be sufficiently large (that is, a sufficient number of buckets are provided), although the bounds for the tour given above are slightly loose. In the next section, we shall give *tight* bounds for the Sierpiński-bucket algorithm.

3. Sierpiński-Bucket Algorithms

In the Sierpiński bucket order, we name buckets from b_1 in the Sierpiński bucket oder as in Fig.3.1. For a pair of buckets b_i, b_j, define $d(b_i, b_j)$ to be an upper bound for the distance (L_∞ or L_2) between two points contained in these two buckets b_i and b_j. For j, define c_j to be $\max_i d(b_i, b_{i+j-1})$. In order to adopt an approach taken in Iri, Murota and Matsui [8] for estimating the worst-case performance of the bucket algorithm, we must first evaluate c_j, which is not so easy in the case of the Sierpiński bucket order. We first consider the case of L_∞ distance and then that of the L_2 distance. We shall only consider the case

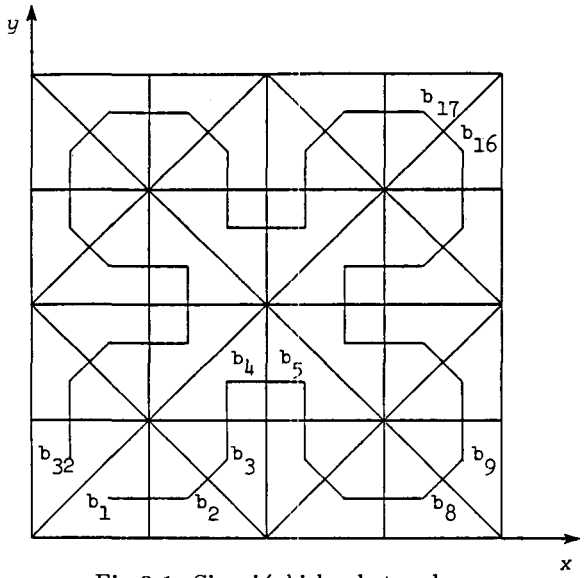


Fig.3.1. Sierpiński bucket order

in which the unit square is divided into k^2 subsquares, and each subsquare is divided into two triangles (the number of triangular buckets is $2k^2$ and $k = 2^i$ for an integer i).

3.1. Worst-case analysis with respect to the L_∞ distance

We trivially have $c_1 = 1/k$, and c_2, $c_3 = 2/k$. However, it is not trivial to evaluate c_j for general j, which we shall first investigate. For two buckets b_i and b_j, a pair (b_i, b_j) is called *congruent* to (b_1, b_l) $(l = j - i + 1)$ if the Sierpiński curve from b_i to b_j is congruent to the Sierpiński curve from b_1 to b_l. For example, (b_3, b_4) is congruent to (b_1, b_2), but (b_2, b_3) is not. Define $F(j)$ by

$$F(1) = 1, \qquad F(j) = \frac{j + 4^J}{3 \cdot 2^{J-1}} \quad \text{where } J = \lceil \log_4 \frac{j}{2} \rceil \quad (j \geq 2). \tag{3.1}$$

It is noted that $F(j)$ is increasing in j.

Theorem 3.1. When the unit square is divided into $2k^2$ triangular buckets,

(i) For $j \geq 2$, $j \neq 2 \cdot 4^i$, $5 \cdot 4^i$ ($i \geq 0$, integer), we have $kc_j \leq F(j-1) < F(j)$.

(ii) For $j = 2 \cdot 4^i$ or $5 \cdot 4^i$, we have $k \cdot d(b_l, b_{l+j-1}) \leq F(j)$ if (b_l, b_{l+j-1}) is congruent to (b_1, b_j), and $k \cdot d(b_l, b_{l+j-1}) \leq F(j-1) < F(j)$, otherwise. □

We provide three lemmas for proving this theorem. We define d_j to be $d(b_1, b_j)$.

Lemma 3.1. For $j = 1$, $2 \cdot 4^i$, $5 \cdot 4^i$, $kd_j \leq F(j)$. For other j, $kd_j \leq F(j-1)$.

Proof: The cases of $j = 1, 2$ are trivial. Suppose that the lemma holds for j with $j \leq 2 \cdot 4^{i-1}$, and consider j with $2 \cdot 4^{i-1} < j \leq 2 \cdot 4^i$. See Fig.3.2.

(i) For j with $2 \cdot 4^{i-1} < j \leq 4^i$: We have $kd_j \leq 2^i = F(2 \cdot 4^{i-1}) \leq F(j-1)$.

(ii) For j with $4^i + 4^I/2 < j \leq 4^i + 4^I$, $I \in \{0, 1, \ldots, i\}$: The case of $I = 0$ is trivial. For other I, from the induction hypothesis, we have $kd_j \leq 2^i + F(j - (4^i + 4^I/2))$. Define

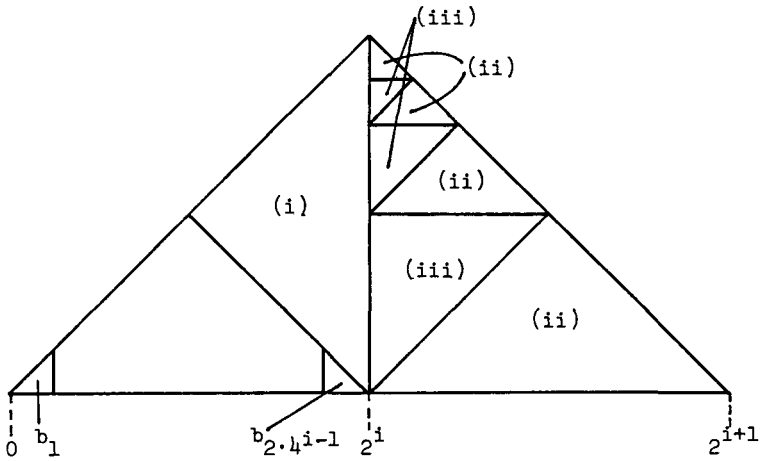


Fig.3.2. Proof of Lemma 3.1

$f(j)$ to be $F(j-1)-(2^i+F(j-4^i-4^I/2))$. As is readily seen, $f(j)$ is nonincreasing in j. We have, for each $I=1,\ldots,i-2$,

$$f(4^i+4^I)=\frac{1}{3\cdot 2^{i-1}}[(2^i-2^I)(2^{i-1}-2^I)-1]>0,$$

and $f(5\cdot 4^{i-1}-1)=1/(3\cdot 2^{i-2})>0$ and $f(2\cdot 4^i-1)=0$, so that, for $j\neq 5\cdot 4^{i-1}$, $2\cdot 4^i$, we have $kd_j\leq F(j-1)$.

For $j=5\cdot 4^{i-1}$, $2\cdot 4^i$, we directly have $kd_j=F(j)$.

(iii) For other j: This case immediately follows from the arguments for (ii). □

Fixing the x- and y-coordinates as in Fig.3.1, we define $\tilde{d}_j$ and $\tilde{F}(j)$ by

$$\tilde{d}_j=\max\{\,y_j\mid \text{point}\,(x_j,y_j)\in b_j\},\tag{3.2}$$

$$\tilde{F}(j)=\frac{j+2\cdot 4^{\lfloor\log_4 j\rfloor}}{3\cdot 2^{\lfloor\log_4 j\rfloor}}\qquad (j\geq 1).\tag{3.3}$$

(e.g, $\tilde{d}_1=\tilde{d}_2=\tilde{d}_3=1/k$, $\tilde{d}_4=2/k$, $\tilde{d}_{13}=3/k$, etc.; note that $\tilde{F}(j)$ is increasing in j.)

Lemma 3.2. For $j\neq 4^i$, $k\tilde{d}_j\leq\tilde{F}(j-1)<\tilde{F}(j)$. For $j=4^i$, $k\tilde{d}_j\leq\tilde{F}(j)$.

Proof: The case of $j=1$ is obvious. Suppose that the lemma holds for j with $j\leq 4^{i-1}$, and consider j with $4^{i-1}<j\leq 4^i$. See Fig.3.3.

(i) For j with $4^{i-1}<j\leq 3\cdot 4^{i-1}$: we have $k\tilde{d}_j\leq 2^{i-1}=\tilde{F}(4^{i-1})\leq\tilde{F}(j-1)$.

(ii) For j with $3\cdot 4^{i-1}<j\leq 4^i$: From the assumption, $k\tilde{d}_j\leq 2^{i-1}+\tilde{F}(j-3\cdot 4^{i-1})$. Define $\tilde{f}(j)$ to be $\tilde{F}(j-1)-(2^{i-1}+\tilde{F}(j-3\cdot 4^{i-1}))$. $\tilde{f}(j)$ is nondecreasing in j, and $\tilde{f}(4^i-1)=0$. Hence, $k\tilde{d}_j\leq\tilde{F}(j-1)$ for j with $3\cdot 4^{i-1}<j<4^i$. The case of $j=4^i$ is directly shown. □

Lemma 3.3. For $j_1,j_2\geq 1$, $\tilde{F}(j_1)+\tilde{F}(j_2)\leq F(j_1+j_2)$.

Proof: We can easily see that, for $j\geq 2$,

$$\max\{\tilde{F}(j_1)+\tilde{F}(j_2)\mid j_1+j_2=j,\ j_1,j_2\geq 1\}=2\tilde{F}(j/2).$$

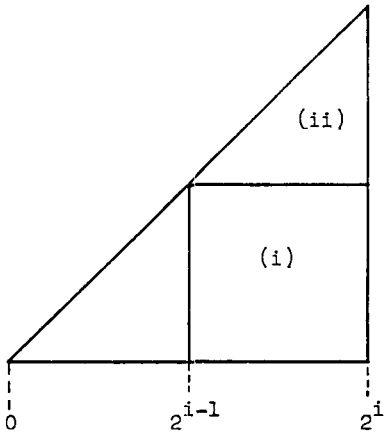


Fig.3.3. Proof of Lemma 3.2

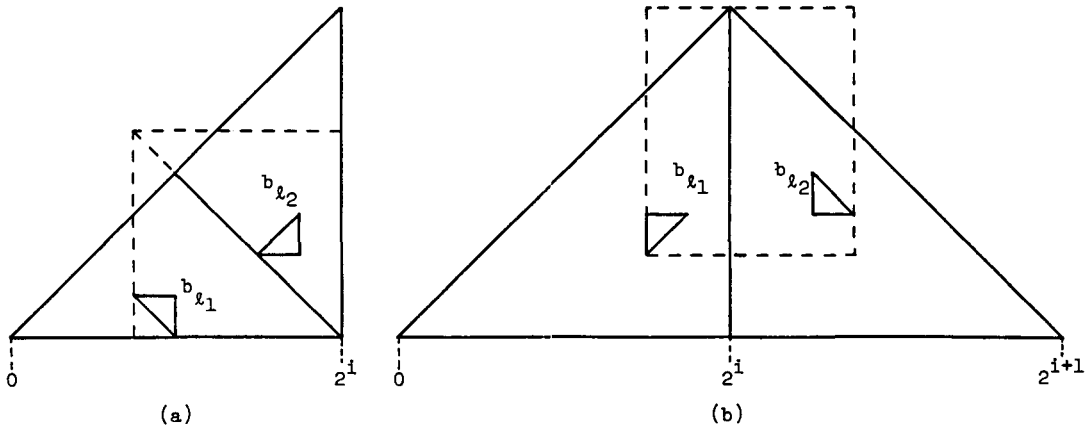


Fig.3.4. Proof of Theorem 3.1

Since $2\tilde{F}(j/2) = F(j)$, we obtain the lemma. □

Now, we shall prove Theorem 3.1.

Proof of Theorem 3.1: (i) The case of $l_1 \leq k^2 < l_2$: We have

$$\begin{aligned} d(b_{l_1}, b_{l_2}) &\leq \max\{d(b_{l_1}, b_{k^2}), d(b_{k^2+1}, b_{l_2})\} \\ &\leq \frac{1}{k}\max\{F(k^2 - l_1 + 1),\ F(l_2 - k^2)\} \leq \frac{1}{k}F(l_2 - l_1). \end{aligned}$$

If the case (i) does not hold, we can assume without loss of generality that $1 \leq l_1 < l_2 \leq k^2$. For such l_1 and l_2, we have only to consider the following two cases owing to the structure of the Sierpiński bucket order (see Fig.3.4).

(ii) The case of $l_1 \leq 2 \cdot 4^{i-1} < l_2 \leq 4^i$ for some i: We have

$$\begin{aligned} d(b_{l_1}, b_{l_2}) &\leq \max\{d(b_{l_1}, b_{2\cdot 4^{i-1}}), d(b_{2\cdot 4^{i-1}+1}, b_{l_2})\} \\ &\leq \frac{1}{k}\max\{F(2\cdot 4^{i-1} - l_1 + 1), F(l_2 - 2\cdot 4^{i-1})\} \leq \frac{1}{k}F(l_2 - l_1). \end{aligned}$$

(iii) The case of $l_1 \le 4^i < l_2 \le 2 \cdot 4^i$ for some i: Let $j_1 = 4^i - l_1 + 1$, $j_2 = l_2 - 4^i$ and $j = j_1 + j_2 (= l_2 - l_1 + 1)$. In this case, we have

$$d(b_{l_1}, b_{l_2}) \le \max\{d_{j_1}, d_{j_2}, \tilde{d}_{j_1} + \tilde{d}_{j_2}\}.$$

If the maximum is attained by d_{j_1} or d_{j_2}, we have $d(b_{l_1}, b_{l_2}) \le \max\{d_{j_1}, d_{j_2}\} \le F(j-1)$.

Suppose that the maximum is attained by $\tilde{d}_{j_1} + \tilde{d}_{j_2}$. In the case of $j_1 \ne 4^{i_1}$, we have $\tilde{d}_{j_1} + \tilde{d}_{j_2} \le \tilde{F}(j_1 - 1) + \tilde{F}(j_2) \le F(j-1)$ from the assumption of the induction. The case of $j_2 \ne 4^{i_2}$ is similar. The remaining case is such that $j_1 = 4^{i_1}$ and $j_2 = 4^{i_2}$. In this case, (b_{l_1}, b_{l_2}) is congruent to (b_1, b_j), and then, from Lemma 3.1, we have $d(b_{l_1}, b_{l_2}) \le F(j)$ if $|i_1 - i_2| \le 1$, and $d(b_{l_1}, b_{l_2}) \le F(j-1)$, otherwise. □

A. With preprocessing and without tour

We shall evaluate $\widehat{M}_n(\alpha)$ where the unit square is divided into $2k^2 \simeq \alpha^2 n$ buckets with $k = 2^{\lceil \log_2 \alpha\sqrt{n} \rceil - 1/2}$. Let n_j be the number of edges in a matching (or in a tour in the analysis with tour) connecting points in two buckets at bucket distance $j-1$, that is, two buckets the difference of ranks of which in the Sierpiński bucket order is $j-1$; in particular, n_1 is the number of pairs within the same bucket. Consider the following linear program and its dual:

$$\begin{aligned} \hat{f}_n(k) \equiv \max \ & \sum_{j=1} \frac{F(j)}{k} n_j \\ \text{s.t.} \quad & \sum_{j=1} n_j = \frac{n}{2} \\ & \sum_{j=2} j n_j \le 2k^2 \\ & n_j \ge 0 \end{aligned} \tag{3.4}$$

and

$$\begin{aligned} \hat{g}_n(k) \equiv \min \ & \frac{n}{2k} x + 2ky \\ \text{s.t.} \quad & x \ge 1 \\ & x + jy \ge F(j) \quad (j = 2, \ldots) \\ & y \ge 0 \end{aligned} \tag{3.5}$$

From Theorem 3.1 and the well-known linear-programming duality theorem,

$$\widehat{M}_n(\sqrt{2}a) \le \hat{f}_n(k) \le \hat{g}_n(k),$$

where $a = k/\sqrt{n}$.

Lemma 3.4. The vertices of the polytope defined by (3.5) in the dual program are $(1, \frac{1}{2})$, $(\frac{4^i}{3 \cdot 2^{i-1}}, \frac{1}{3 \cdot 2^{i-1}})$ $(i = 1, 2, \ldots)$.

Proof: Immediate from the fact that, for j with $2\cdot 4^{i-1} \le j \le 2\cdot 4^i$ and $i \ge 1$, we have

$$\frac{4^i}{3\cdot 2^{i-1}} + j\frac{1}{3\cdot 2^{i-1}} = F(j). \qquad \square$$

Lemma 3.5. For $k = a\sqrt{n}$ $(a > 0)$, $\hat{g}_n(a\sqrt{n})$ is given by

$$\begin{aligned}\hat{g}_n(a\sqrt{n}) &= \sqrt{n}\cdot\min\{\frac{1}{2a}+a,\ \frac{1}{3\cdot 2^{i-1}}(\frac{4^i}{2a}+2a)\ (i=1,\ldots)\}\\ &= \begin{cases}\sqrt{n}(\frac{1}{2a}+a) & (0 < a \le \frac{1}{\sqrt{2}})\\ \sqrt{n}\frac{1}{3\cdot 2^{i-1}}(\frac{4^i}{2a}+2a) & (\frac{2^{i-1}}{\sqrt{2}} \le a \le \frac{2^i}{\sqrt{2}})\quad (i=1,\ldots)\end{cases}\end{aligned}$$

Proof: The first equality follows from Lemma 3.4. The second follows from the following inequalities, where $h(i) \equiv \frac{1}{3\cdot 2^{i-1}}(\frac{4^i}{2a}+2a)$:

$$h(i) \lesseqgtr h(i+1) \quad \text{iff} \quad a \lesseqgtr \frac{2^i}{\sqrt{2}}, \qquad \frac{1}{2a}+a \lesseqgtr h(1) \quad \text{iff} \quad a \lesseqgtr \frac{1}{\sqrt{2}}. \qquad \square$$

Since $k = 2^{\lceil \log_2 \alpha\sqrt{n}\rceil - 1/2}$ and $k = a\sqrt{n}$, we have $\alpha \le \sqrt{2}a < 2\alpha$, and hence

$$\hat{\mu}_O(\alpha) = \lim_{n\to\infty}(\sup_{\alpha\le\sqrt{2}a<2\alpha} \widehat{M}_n(\sqrt{2}a)/\sqrt{n}).$$

Then, we have the following theorem.

Theorem 3.2. With respect to the L_∞ distance, for the Sierpiński-bucket algorithm, for matchings, with preprocessing and without tour, we have

$$\hat{\mu}_O(\alpha) = \begin{cases}\sqrt{2}(\frac{\alpha}{2}+\frac{1}{2\alpha}) & (0 < \alpha \le 1)\\ \sqrt{2} & (\alpha \ge 1)\end{cases} \qquad \hat{\mu}_O = \sqrt{2}.$$

Proof: From Lemma 3.5, we readily see that

$$\sup_{\alpha\le\sqrt{2}a<2\alpha} \widehat{M}_n(\sqrt{2}a) \le \begin{cases}\sqrt{2}(\frac{\alpha}{2}+\frac{1}{2\alpha}) & (0 < \alpha \le 1)\\ \sqrt{2} & (\alpha \ge 1)\end{cases}$$

Consider the lower bound of $\hat{\mu}_O(\alpha)$. In the case of $0 < \alpha \le 1$, we can easily construct a configuration of $n(i)$ points in the unit square with $2k(i)^2$ buckets such that $n(i) = \lceil 2\cdot 4^i/\alpha^2\rceil$, $k(i) = 2^i$, $n_2 = k(i)^2$, $n_1 = n(i)/2 - k(i)^2$ and $n_j = 0$ $(j \ne 1, 2)$. For this set of points, the cost of the matching is $\frac{2}{k(i)}n_2 + \frac{1}{k(i)}n_1 \simeq \sqrt{2}(\frac{\alpha}{2}+\frac{1}{2\alpha})\sqrt{n(i)}$.

In the case of $\alpha \ge 1$, we can construct a configuration of $n(i)$ points in the unit square with $2k(i)^2$ buckets such that $n(i) = \lceil 2\cdot 4^i\rceil$, $k(i) = 2^{i+a}$ with $a = \lceil \log_2 \alpha\rceil$ and $n_{2\cdot 4^a} = \frac{n(i)}{2}$, $n_j = 0$ $(j \ne 2\cdot 4^a)$. For this set of points, the cost of the matching is $\frac{2^{a+1}}{k(i)}\frac{n(i)}{2} = 2^{i+1} = \sqrt{2n(i)}$. $\square$

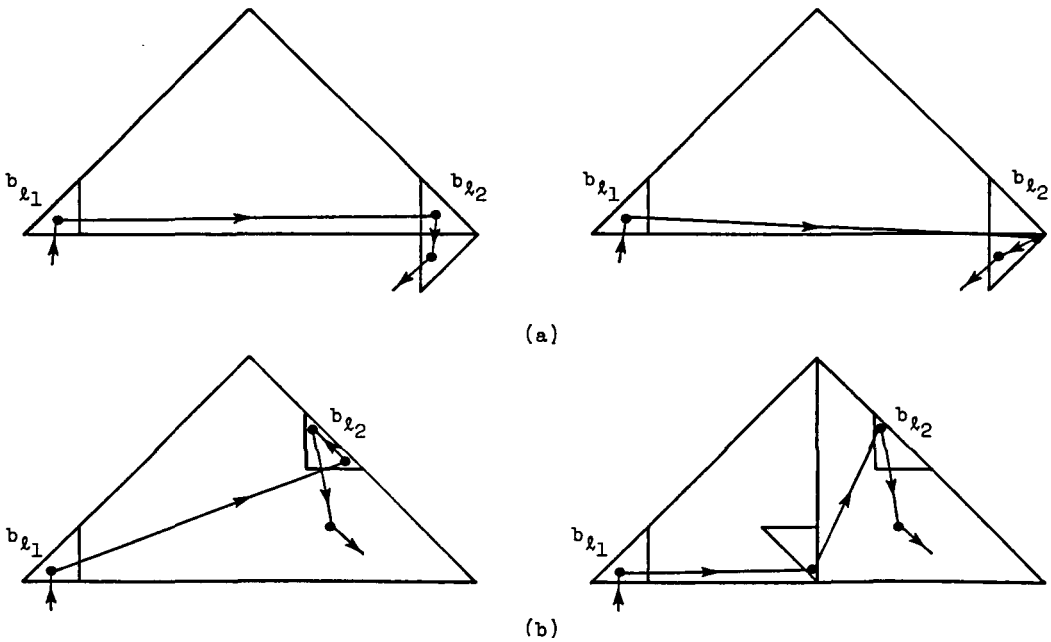


Fig.3.5. Proof of Lemma 3.6

B. Without preprocessing and with tour

In this case, we have only to evaluate $\widehat{T}_n(\alpha)$, since $\widehat{M}_n(\alpha) \leq \widehat{T}_n(\alpha)/2$. Concerning $\widehat{T}_n(\alpha)$, first consider the following lemma.

Lemma 3.6. For any tour T constructed by the Sierpiński-bucket algorithm, there is a tour T' constructed by this algorithm such that

(i) no edge connects points in buckets b_{l_1} and b_{l_2} with $l_2 - l_1 + 1 = 2 \cdot 4^i$ or $5 \cdot 4^i$ such that (b_{l_1}, b_{l_2}) is congruent to $(b_1, b_{l_2 - l_1 + 1})$;

(ii) (the length of T') $\geq$(the length of T).

Proof: Starting from T, for each edge not satisfying (i), we move points in bucket b_{l_2} to bucket b_{l_2+1} (except the case of Fig.3.5(b)) as depicted in Fig.3.5 until there comes to be no such an edge. (There are many cases that must be considered, but only some of them are depicted in Fig.3.5.) We can execute this procedure so that it halts in finite steps, and each step does not decrease the length of the tour. Hence, we obtain the lemma. □

Consider the following linear program and its dual:

$$\begin{aligned} \hat{f}_n(k) \equiv \max \ & \frac{n_1}{k} + \sum_{j=2} \frac{F(j-1)}{k} n_j \\ \text{s.t.} \quad & \sum_{j=1} n_j = n \\ & \sum_{j=2} (j-1) n_j \leq 2k^2 \\ & n_j \geq 0 \end{aligned} \tag{3.6}$$

and

$$\begin{aligned} \hat{g}_n(k) \equiv \min\ & \frac{n}{k}x + 2ky \\ \text{s.t.}\quad & x \geq 1 \\ & x + jy \geq F(j) \quad (j = 1, 2, \ldots) \\ & y \geq 0 \end{aligned} \tag{3.7}$$

From Theorem 3.1 and Lemma 3.6, we have

$$\hat{T}_n(\sqrt{2}a) \leq \hat{f}_n(k) \leq \hat{g}_n(k),$$

where $a = k/\sqrt{n}$.

Theorem 3.3. With respect to the L_∞ distance, for the Sierpiński-bucket algorithm for tours, we have

$$\hat{\tau}_{\mathrm{O}}(\alpha) = \begin{cases} \sqrt{2}(\frac{\alpha}{2} + \frac{1}{\alpha}) & (\alpha \leq \sqrt{2}) \\ 2 & (\alpha \geq \sqrt{2}) \end{cases} \qquad \hat{\tau}_{\mathrm{O}} = 2.$$

Proof: From Lemma 3.4, we have

$$\hat{g}_n(a\sqrt{n}) = \min\{(a + \frac{1}{a}),\ \frac{1}{3 \cdot 2^{i-1}}(\frac{4^i}{a} + 2a)\ (i = 1, 2, \ldots)\}.$$

Letting $h(i) \equiv \frac{1}{3 \cdot 2^{i-1}}(\frac{4^i}{a} + 2a)$, we have

$$h(i) \lesseqgtr h(i+1) \quad \text{iff } a \lesseqgtr 2^i, \qquad a + \frac{1}{a} \lesseqgtr h(1) \quad \text{iff } a \lesseqgtr 1.$$

Hence, we have

$$\hat{g}_n(a\sqrt{n}) = \begin{cases} a + \frac{1}{a} & (0 < a \leq 1) \\ \frac{1}{3 \cdot 2^{i-1}}(\frac{4^i}{a} + 2a) & (2^{i-1} \leq a \leq 2^i) \end{cases}$$

and

$$\sup_{\alpha \leq \sqrt{2}a < 2\alpha} \hat{T}_n(\sqrt{2}a) \leq \begin{cases} \sqrt{2}(\frac{\alpha}{2} + \frac{1}{\alpha}) & (0 < \alpha \leq \sqrt{2}) \\ 2 & (\alpha \geq \sqrt{2}) \end{cases}$$

The lower bound can be shown similarly as in Theorem 3.2. □

Corollary 3.1. With respect to the L_∞ distance, for the Sierpiński-bucket algorithm, for matchings, without preprocessing and with tour, we have

$$\hat{\mu}_{\mathrm{O}}(\alpha) = \begin{cases} \sqrt{2}(\frac{\alpha}{4} + \frac{1}{2\alpha}) & (\alpha \leq \sqrt{2}) \\ 1 & (\alpha \geq \sqrt{2}) \end{cases} \qquad \hat{\mu}_{\mathrm{O}} = 1. \quad \square$$

3.2. Worst-case analysis with respect to the L_2 distance

In this case, from the proof of Theorem 3.1 in Platzman and Bartholdi [12], we can obtain the following.

Lemma 3.7. $c_j \le \sqrt{2j}/k$. □

Using this lemma, we can analyze $\hat{\mu}_O(\alpha)$ for the L_2 distance by similar techniques as above, although, in this case, the linear programs are harder to solve directly, so that we solve it by allowing the values of j to be continuous.

A. With preprocessing and without tour

As in the L_∞ case, consider the following linear program and its dual:

$$\begin{aligned} \hat{f}_n(k) \equiv \max \ & \sum_{j=1} \frac{\sqrt{2j}}{k} n_j \\ \text{s.t.} \quad & \sum_{j=1} n_j = \frac{n}{2} \\ & \sum_{j=2} j n_j \le 2k^2 \\ & n_j \ge 0 \end{aligned} \tag{3.8}$$

and

$$\begin{aligned} \hat{g}_n(k) \equiv \min \ & \frac{n}{2k} x + 2ky \\ \text{s.t.} \quad & x \ge \sqrt{2} \\ & x + jy \ge \sqrt{2j} \quad (j = 2, \ldots) \\ & y \ge 0 \end{aligned} \tag{3.9}$$

In this dual program, replace j by real number z with $z \ge 2$:

$$\begin{aligned} \tilde{g}_n(k) \equiv \min \ & \frac{n}{2k} x + 2ky \\ \text{s.t.} \quad & x \ge \sqrt{2} \\ & x + zy \ge \sqrt{2z} \quad \text{for any } z \ge 2 \\ & y \ge 0 \end{aligned} \tag{3.10}$$

We have $\widehat{M}_n(\sqrt{2}a) \le \hat{f}_n(k) \le \hat{g}_n(k) \le \tilde{g}_n(k)$, where $k = a\sqrt{n}$. We can compute $\tilde{g}_n(a\sqrt{n})$ directly:

$$\tilde{g}_n(a\sqrt{n}) = \begin{cases} \sqrt{2}(\frac{a}{2} + \frac{1}{2a}) & (0 < a \le 1) \\ \sqrt{2} & (a \ge 1) \end{cases} \tag{3.11}$$

Hence, we obtain the following.

Theorem 3.4. With respect to the L_2 distance, for the Sierpiński-bucket algorithm, for matchings, with preprocessing and without tour, we have

$$\hat{\mu}_O(\alpha) = \begin{cases} \frac{\alpha}{2} + \frac{1}{\alpha} & (0 < \alpha \le \sqrt{2}) \\ \sqrt{2} & (\alpha \ge \sqrt{2}) \end{cases} \qquad \hat{\mu}_O = \sqrt{2}.$$

Proof: The upper bound of $\hat{\mu}_O(\alpha)$ is obtained from (3.11). Concerning the lower bound, in the case of $0 < \alpha \le \sqrt{2}$, we can construct a configuration of $n(i)$ points in the unit square with $2k(i)^2$ buckets such that $n(i) = \lceil 2 \cdot 4^i/\alpha^2 \rceil$, $k(i) = 2^i$, $n_4 = k(i)^2/2$ and $n_1 = n(i)/2 - k(i)^2/2$. For this set of points, the lengths of the matching is $\frac{2\sqrt{2}}{k(i)} n_4 + \frac{\sqrt{2}}{k(i)} n_1 \simeq (\frac{\alpha}{2} + \frac{1}{\alpha})\sqrt{n(i)}$. The case of $\alpha \ge \sqrt{2}$ is easy. □

B. Without preprocessing and with tour

Concerning $\hat{T}_n(\sqrt{2}a)$ with $k = a\sqrt{n}$, consider the following linear program and its dual:

$$\begin{aligned} \hat{f}_n(k) \equiv \max \sum_{j=1} \frac{\sqrt{2j}}{k} n_j & \\ \text{s.t.} \quad \sum_{j=1} n_j = n & \\ \sum_{j=2} (j-1) n_j \le 2k^2 & \\ n_j \ge 0 & \end{aligned} \tag{3.12}$$

and

$$\begin{aligned} \hat{g}_n(k) \equiv \min \frac{n}{k} x + 2ky & \\ \text{s.t.} \quad x + (j-1)y \ge \sqrt{2j} \quad (j = 1, 2, \ldots) & \\ y \ge 0 & \end{aligned} \tag{3.13}$$

In this dual program, replace j by real number z with $z \ge 1$:

$$\begin{aligned} \tilde{g}_n(a\sqrt{n}) \equiv \min \sqrt{n}(\frac{x}{a} + 2ay) & \\ \text{s.t.} \quad x + (z-1)y \ge \sqrt{2z} \quad \text{for any } z \ge 1 & \\ y \ge 0 & \end{aligned} \tag{3.14}$$

$\tilde{g}_n(a\sqrt{n})$ is given by $\tilde{g}_n(a\sqrt{n}) = 2\sqrt{1 + 1/(2a^2)}\sqrt{n}$. Then, applying Lemma 2.1, we have the following.

Theorem 3.5. With respect to the L_2 distance, for the Sierpiński-bucket algorithm for tours, we have

$$\hat{\tau}_O(\alpha) \begin{cases} \le 2\sqrt{1 + 1/\alpha^2} & (0 < \alpha \le 2\sqrt{2}) \\ \le \frac{\alpha}{4} + \frac{4}{\alpha} & (2\sqrt{2} \le \alpha \le 4) \\ = 2 & (\alpha \ge 4) \end{cases}, \qquad \hat{\tau}_O = 2.$$

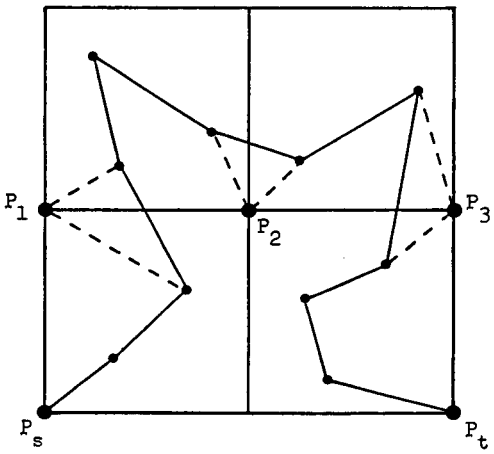


Fig.4.1. Proof of Lemma 4.2

(Note that, for the Sierpiński-bucket algorithm, for matchings, without preprocess and with tour, $\hat{\mu}_O(\alpha) = \hat{\tau}_O(\alpha)/2$.) □

4. Hilbert-Curve Algorithms

Let $H(n)$ be the supremum of the costs of paths starting from the initial point, terminating at the last point of the Hilbert curve and connecting arbitrary n points in the unit square which are constructed by the Hilbert-curve algorithm. By definition, we have the following.

Lemma 4.1. For the Hilbert-curve algorithm, we have $\hat{T}_n \leq H(n) + 1$. □

In the sequel, we shall evaluate $H(n)$.

Lemma 4.2. $H(n)$ satisfies the following.

$$H(n) \leq \max\{\sum_{i=1}^{4} \frac{1}{2} H(n_i) \mid \sum_{i=1}^{4} n_i = n,\ n_i \geq 0,\ \text{integers}\}. \tag{4.1}$$

Proof: For any path obtained by the Hilbert-curve algorithm starting from the initial point P_s and terminating at the last point P_t, consider a path obtained from that path by dropping at each point P_1, P_2, P_3 as in Fig.4.1. Then, the lemma immediately follows from the triangle inequality of the distance. □

4.1. Worst-case analysis with respect to the L_∞ distance

$H(0)$, $H(1)$ and $H(2)$ can be easily computed, and then, using (4.1), we have

$$H(0) = 1, \quad H(1) = 2, \quad H(2) = 3 \quad \text{and} \quad H(3) = 7/2. \tag{4.2}$$

Lemma 4.3. For n with $3 \leq n \leq 16$, we have

$$H(n) \leq \frac{7\sqrt{3}}{6}\sqrt{n}. \tag{4.3}$$

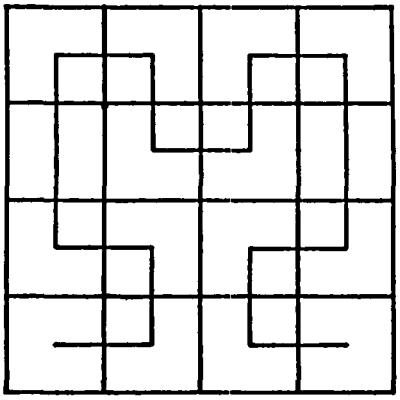


Fig.4.2. Hilbert bucket order of 4×4 square buckets

Proof: Since $H(3) = 7/2$ as obtained in (4.2), we consider n with $4 \leq n \leq 16$. In order to obtain the lemma, we again adopt the linear-programming approach. Consider the partition of the unit square into 4×4 square buckets as depicted in Fig.4.2. For any $n+2$ points distributed in this unit square, make a path connecting those points in the Hilbert bucket order, where points in the same bucket are connected arbitrarily. Let $h(n+2)$ be the supremum of lengths of such paths over all possible configurations of $n+2$ points. By definition, we have $H(n) \leq h(n+2)$.

Concerning the set of edges of a path thus obtained, define n_i $(i = 1, \ldots, 6)$ to be the numbers of edges connecting points in two buckets at bucket distance $i-1$, that is, two buckets the difference of ranks of which in the Hilbert bucket order is $i-1$ (for $i = 6$, bucket distance at least 5). For example, n_1 is the number of edges connecting a pair of points in the same bucket. For $i = 3$ and 5, we separately count the numbers of edges, as depicted in Fig.4.3, and denote them by n_3' and n_5', respectively. Define $\hat{f}(n+2)$ by

$$\begin{aligned} \hat{f}(n+2) \equiv \max\ & n_1 + 2n_2 + 2n_3 + 3n_3' + 3n_4 + 3n_5 + 4n_5' + 4n_6 \\ \text{s.t.}\quad & n_1 + n_2 + n_3 + n_3' + n_4 + n_5 + n_5' + n_6 = n+1 \\ & n_2 + 2(n_3 + n_3') + 3n_4 + 4(n_5 + n_5') + 5n_6 \leq 15 \\ & n_3' + n_5' \leq 2 \\ & n_i, n_i' \geq 0 \end{aligned} \tag{4.4}$$

In the above linear program, variables n_3 and n_5 can be set to be 0 in order to obtain the maximum, and then its dual is described as follows:

$$\begin{aligned} \hat{g}(n+2) \equiv \min\ & (n+1)x + 15y + 2z \\ \text{s.t.}\quad & x \geq 1 \\ & x + y \geq 2 \\ & x + 2y + z \geq 3 \\ & x + 3y \geq 3 \end{aligned} \tag{4.5}$$

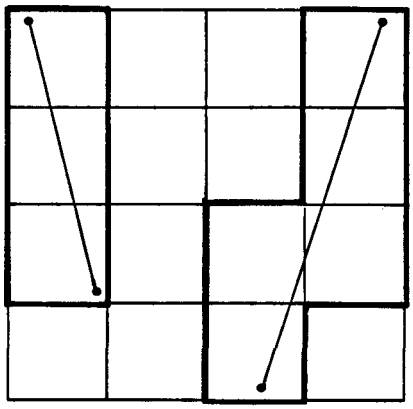


Fig.4.3. Edges counted by n_3' and n_5'

$$\begin{aligned} x + 4y + z &\geq 4 \\ x + 5y &\geq 4 \\ y, z &\geq 0 \end{aligned}$$

Then, we have

$$H(n) \leq h(n+2) \leq \frac{1}{4}\hat{f}(n+2) \leq \frac{1}{4}\hat{g}(n+2).$$

Concerning the polytope defined by (4.5), its vertices are (1,1,0), $(\frac{3}{2}, \frac{1}{2}, \frac{1}{2})$, $(2, \frac{1}{2}, 0)$ and (4,0,0), and hence

$$\frac{1}{4}\hat{g}(n+2) = \begin{cases} \frac{1}{4}(\frac{3}{2}n + 10) \leq \frac{7\sqrt{3}}{6}\sqrt{n} & (4 \leq n \leq 12) \\ \frac{1}{4}(n + 16) \leq \frac{7\sqrt{3}}{6}\sqrt{n} & (12 \leq n \leq 16) \end{cases} \tag{4.6}$$

Thus, we obtain the lemma. □

Lemma 4.4. $H(n) \leq \frac{7\sqrt{3}}{6}\sqrt{n}$ $(n \geq 3)$.

Proof: From Lemma 4.3, we have only to consider $n \geq 17$. Suppose that $H(n') \leq \frac{7\sqrt{3}}{6}\sqrt{n'}$ for any n' with $3 \leq n' < n$ and $n \geq 17$. For $n \geq 17$, if the maximum in (4.1) is attained by $n_i = n$, $n_j = 0$ $(j \neq i)$ for some i, $H(n) \leq 3 < \frac{7\sqrt{6}}{6}\sqrt{n}$. Let n_i $(i = 1, 2, 3, 4)$ be integers such that $\sum_{i=1}^{4} n_i = n$ and $0 \leq n_i < n$. For n_i, define $s(n_i)$ by $s(n_i) = \max\{3, n_i\} - n_i$, where $s(n_i) = 0, 1, 2$ or 3 and $3 \leq n_i + s(n_i) < n$, and let $s \equiv \sum_{i=1}^{4} s(n_i)$. Since $H(l) \leq H(3) - \frac{3-l}{2}$ for $l = 0, 1, 2$, we have

$$\begin{aligned} \frac{1}{2}\sum_{i=1}^{4} H(n_i) &\leq \frac{1}{2}\sum_{i=1}^{4}(H(n_i + s(n_i)) - \frac{s(n_i)}{2}) \leq \frac{1}{2}\sum_{i=1}^{4}(\frac{7\sqrt{3}}{6}\sqrt{n_i + s(n_i)} - \frac{s(n_i)}{2}) \\ &\leq \frac{7\sqrt{3}}{6}\sqrt{n+s} - \frac{s}{4} \leq \frac{7\sqrt{3}}{6}(\sqrt{n} + \frac{s}{2\sqrt{n}}) - \frac{s}{4} \\ &\leq \frac{7\sqrt{3}}{6}\sqrt{n}, \end{aligned}$$

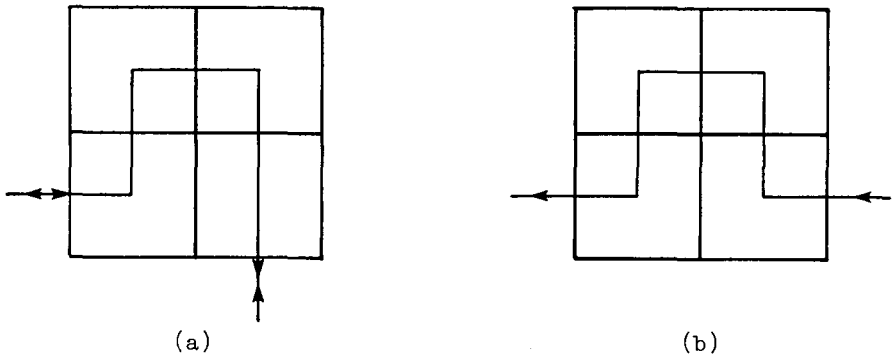


Fig.4.4. Types (a) and (b) of four buckets

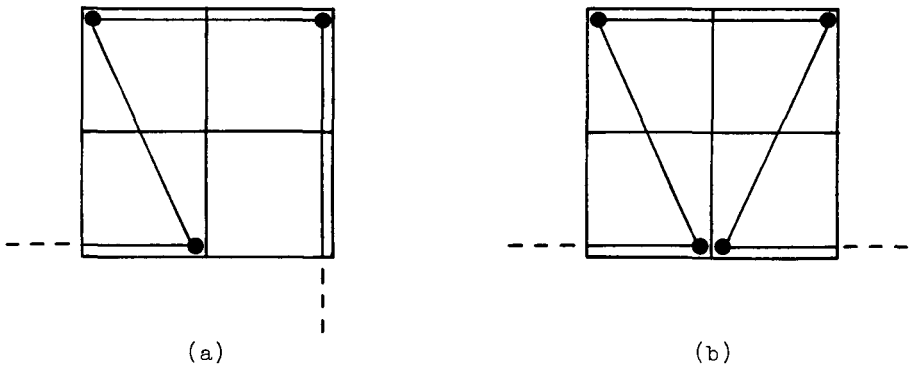


Fig.4.5. Location of points for four buckets

where the last inequality follows from $\frac{7\sqrt{3}}{12\sqrt{n}} - \frac{1}{4} < 0$ for $n \geq 17$. □

We now consider a lower bound of $\hat{\mu}_O$. Consider the Hilbert bucket order H_i in the unit square with k^2 buckets ($k = 2^i$), where the buckets are named from b_1 to b_{k^2} in the Hilbert bucket order. For each $j = 1, \ldots, k^2/4$, four buckets b_{4j-3}, b_{4j-2}, b_{4j-1}, b_{4j} can be classified by (a) and (b) as depicted in Fig.4.4. For four buckets of type (a), locate three points as in Fig.4.5(a), and, for four buckets of type (b), locate four points as in Fig.4.5(b). We consider four buckets with $j = 1$, $k^2/4$ are of type (b). Let $n(i)$ be the number of points totally located in the unit square, $a(i)$ and $b(i)$ be the numbers of four buckets of type (a) and (b), respectively. Then, we have

$$n(i) = 3a(i) + 4b(i),$$

$$a(i) + b(i) = 4^{i-1}.$$

Concerning $b(i)$, we can easily see the following:

$$b(2j+1) = 4b(2j) - 4, \qquad b(2j+2) = 4b(2j+1) - 2,$$

$$b(1) = 1, \qquad b(2) = 2.$$

Hence, we have

$$b(2j+1) = \frac{4+16^j}{5},$$

$$b(2j+2) = \frac{6+4\cdot 16^j}{5}.$$

That is, $a(i) \simeq 4^i/5$, $b(i) \simeq 4^{i-1}/5$, $n(i) \simeq 4^{i+1}/5$, and the length of a path connecting those points in the Hilbert bucket order is $\frac{7}{2^i}a(i) + \frac{8}{2^i}b(i) - \frac{2}{2^i} \simeq \frac{9}{5}2^i \simeq \frac{9\sqrt{5}}{10}\sqrt{n(i)}$. Hence, we have the following theorem.

Theorem 3.6. With respect to the L_∞ distance, for the Hilbert-curve algorithm, we have

$$2.012 \simeq \frac{9\sqrt{5}}{10} \le \hat{\tau}_O \le \frac{7\sqrt{3}}{6} \simeq 2.021. \quad \square$$

4.2. Worst-case analysis with respect to the L_2 distance

In the case of the L_2 distance, we can estimate $\hat{\tau}_O$ in a similar but more hard way. In this case, $H(0) = 1$, $H(1) = 1+\sqrt{2}$. We see $H(2) = \sqrt{2} + \frac{\sqrt{5}}{2} + \frac{1}{2}$. For $H(n)$ with $3 \le n \le 9$, by considering 4×4 buckets as in the L_∞ case, we have $H(n) \le \hat{f}(n)/4$ where

$$\begin{aligned}
\hat{f}(n) \equiv \max\ & \sqrt{2}n_1 + \sqrt{5}n_2 + \sqrt{8}n_3 + \sqrt{10}n_3' + \sqrt{13}n_4' \\
& + \sqrt{10}n_5 + \sqrt{20}n_5' + \sqrt{17}n_6 + \sqrt{20}n_7' + 5n_8' + \sqrt{32}n_{11}' \\
\text{s.t.}\quad & \sum_j (n_j + n_j') = n+1 \qquad (4.7)\\
& \sum_j (j-1)(n_j + n_j') \le 15 \\
& n_3' + n_4' + n_5' + n_7' + n_8' + n_{11}' \le 2 \\
& n_j,\ n_j' \ge 0 \\
& n_6 \le 1 \quad \text{in the case of } n=3 \\
& n_5 + n_7' + n_8' + n_{11}' \le 0 \quad \text{in the case of } n=3 \\
& n_3 + n_5' + n_6 \le 3 \quad \text{in the case of } n=3,4 \\
& n_5' \le 1 \quad \text{in the case of } n=9
\end{aligned}$$

Lemma 4.5. For n with $2 \le n \le 9$, $H(n) \le \frac{\sqrt{5}+2\sqrt{20}+\sqrt{17}}{4\sqrt{3}}\sqrt{n}$.

Proof: The case of $n=2$ is directly shown. For n with $3 \le n \le 9$, an optimum solution of $\hat{f}(n)$ for each n is as follows, where we only show the values of nonzero variables.

$n=3:\ n_2 = 1,\ n_5' = 2,\ n_6 = 1;$ $\qquad$ $n=4:\ n_2 = 2,\ n_5' = 2,\ n_6 = 1;$

$n=5:\ n_2 = 1,\ n_3 = 3,\ n_5' = 2;$ $\qquad$ $n=6:\ n_2 = 3,\ n_3 = 2,\ n_5' = 2;$

$n=7:\ n_2 = 5,\ n_3 = 1,\ n_5' = 2;$ $\qquad$ $n=8:\ n_2 = 7,\ n_5' = 2;$

$n=9:\ n_2 = 7,\ n_3 = 1,\ n_3' = 1,\ n_5' = 1;$

$\hat{f}(3) = \frac{\sqrt{5}+2\sqrt{20}+\sqrt{17}}{\sqrt{3}}\sqrt{3}$, and $\hat{f}(n) < \frac{\sqrt{5}+2\sqrt{20}+\sqrt{17}}{\sqrt{3}}\sqrt{n}$ for n with $4 \le n \le 9$, and we obtain the lemma. □

Using these values, we have the following.

Lemma 4.6. We have $H(n) \le \frac{\sqrt{5}+2\sqrt{20}+\sqrt{17}}{4\sqrt{3}}\sqrt{n}$ $(n \ge 2)$.

Proof: From lemma 4.5, we have only to consider $n \ge 10$. Suppose that the lemma holds for n' with $2 \le n' < n$, and consider $H(n)$. For $n \ge 10$, the maximum in (4.1) is not attained by $n_i = n$, $n_j = 0$ $(j \ne i)$ for i. Let n_i $(i = 1,2,3,4)$ be integers such that $\sum_{i=1}^{4} n_i = n$ and $0 \le n_i < n$. For n_i, define $s(n_i)$ by $s(n_i) = \max\{2, n_i\} - n_i$, and let $s \equiv \sum_{i=1}^{4} s(n_i)$. Then, we have

$$\begin{aligned}
\frac{1}{2}\sum_{i=1}^{4} H(n_i) &\le \frac{1}{2}\sum_{i=1}^{4}\Big(\frac{\sqrt{5}+2\sqrt{20}+\sqrt{17}}{4\sqrt{3}}\sqrt{n_i+s(n_i)} \\
&\qquad -\Big(\frac{\sqrt{5}+2\sqrt{20}+\sqrt{17}}{4\sqrt{3}}\sqrt{2}-1-\sqrt{2}\Big)s(n_i)\Big) \\
&\le \frac{\sqrt{5}+2\sqrt{20}+\sqrt{17}}{4\sqrt{3}}\sqrt{n+s} - \frac{s}{2}\Big(\frac{\sqrt{5}+2\sqrt{20}+\sqrt{17}}{4\sqrt{3}}\sqrt{2}-1-\sqrt{2}\Big) \\
&\le \frac{\sqrt{5}+2\sqrt{20}+\sqrt{17}}{4\sqrt{3}}\sqrt{n} + \frac{s}{2}\Big(\frac{\sqrt{5}+2\sqrt{20}+\sqrt{17}}{4\sqrt{3}}\Big(\frac{1}{\sqrt{n}}-\sqrt{2}\Big)+1+\sqrt{2}\Big) \\
&\le \frac{\sqrt{5}+2\sqrt{20}+\sqrt{17}}{4\sqrt{3}}\sqrt{n}. \qquad \square
\end{aligned}$$

Concerning a lower bound, for the same configuration of $n(i)$ points as in the L_∞ case, the length of a path connecting those points in the Hilbert bucket order is

$$\frac{5+\sqrt{5}}{2^i}a(i) + \frac{4+2\sqrt{5}}{2^i}b(i) - \frac{2}{2^i} \simeq \frac{15+12\sqrt{5}}{20}\sqrt{n(i)},$$

and we obtain the following theorem.

Theorem 4.7. With respect to the L_2 distance for the Hilbert-curve algorithm, we have

$$2.092 \simeq \frac{15+12\sqrt{5}}{20} \le \hat{\tau}_O \le \frac{\sqrt{5}+2\sqrt{20}+\sqrt{17}}{4\sqrt{3}} \simeq 2.209. \qquad \square$$

5. Concluding Remarks

We can consider a "Hilbert-bucket algorithm" in a way similar to the case for the Sierpiński-bucket algorithm and the Sierpiński-curve algorithm. For this algorithm, we

can easily (but a little loosely) evaluate the worst-case performance by means of Lemma 2.1, since we have analyzed $\hat{\tau}_O$ for the Hilbert-curve algorithm.

Concerning the average-case performance of the algorithms considered in this paper, computational experiments by Sanae [5] (see also [1]) for uniformly distributed n points in the unit square suggest that, concerning matchings, the Sierpiński-bucket algorithm may be a little better than the spiral-rack algorithm in [8]. It would be interesting to analyze the average-case performance of these algorithm theoretically as in [8].

Acknowledgment

The author would like to thank Professor Masao Iri of the University of Tokyo for his valuable discussions and suggestions on the problems considered in this paper. This work was partially supported by the Grant-in-Aid for Encouragement of Young Scientists of the Ministry of Education, Science and Culture of Japan under Grant: (A) 60790046 (1985). The author was also supported by the Iue Memorial Foundation.

References

[1] Asano, T., Edahiro, M., Imai, H., Iri, M., and Murota, K.: Practical Use of Bucketing Techniques in Computational Geometry. In "*Computational Geometry*" (G. T. Toussaint, ed.), North-Holland, Amsterdam, 1985, 153–195.

[2] Avis, D.: A Survey of Heuristics for the Weighted Matching Problem. *Networks*, Vol.13 (1983), 475–493.

[3] Bartholdi, J. J., III, and Platzman, L. K.: An $O(N \log N)$ Planar Travelling Salesman Heuristic Based on Spacefilling Curves. *Operations Research Letters*, Vol.1, No.4 (1982), 121–125.

[4] Bartholdi, J. J., III, and Platzman, L. K.: A Fast Heuristic Based on Spacefilling Curves for Minimum-Weight Matching in the Plane. *Information Processing Letters*, Vol.17 (1983), 177–180.

[5] Imai, H., Sanae, H., and Iri, M.: A Planar-Matching Heuristic by Means of Triangular Buckets (in Japanese). *Proceedings of the 1984 Fall Conference of the Operations Research Society of Japan*, 2-D-4, 157–158.

[6] Iri, M., Murota, K., and Matsui, S.: Linear-Time Approximation Algorithms for Finding the Minimum-Weight Perfect Matching on a Plane. *Information Processing Letters*, Vol.12 (1981), 206–209.

[7] Iri, M., Murota, K., and Matsui, S.: An Approximate Solution for the Problem of Optimizing the Plotter Pen Movement. In "*System Modeling and Optimization*" (Proceedings of the 10th IFIP Conference on System Modeling and Optimization, New York, 1981; R. F. Drenick and F. Kozin, eds.), Lecture Notes in Control and

Information Sciences 38, Springer-Verlag, Berlin, 1982, 572–580.

[8] Iri, M., Murota, K., and Matsui, S.: Heuristics for Planar Minimum-Weight Perfect Matchings. *Networks*, Vol.13 (1983), 67–92.

[9] Lawler, E. L.: *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart and Winston, New York, 1976.

[10] Ohya, T., Iri, M., and Murota, K.: Improvements of the Incremental Method for the Voronoi Diagram with Computational Comparison of Various Algorithms. *Journal of the Operations Research Society of Japan*, Vol.27, No.4 (1984), 306–336.

[11] Papadimitriou, C. H.: The Euclidean Travelling Salesman Problem is NP-Complete. *Theoretical Computer Science*, Vol.4 (1977), 237–244.

[12] Platzman, L. K., and Bartholdi, J. J., III: Spacefilling Curves and Routing Problems in the Plane. *PDRC Report Series 83-02*, School of Industrial and Systems Engineering, Georgia Institute of Technology, 1983.

Hiroshi IMAI: Department of Mathematical Engineering and Instrumentation Physics, Faculty of Engineering, University of Tokyo, Bunkyo-ku, Tokyo, Japan 113