# 平面マッチング問題の近似解法

室田　一雄

## 1. はじめに

計算幾何学（computational geometry）という言葉もどうやら正統な述語として定着してきたようである．これは，幾何学的な図形の処理を，計算複雑度(computational complexity)の観点から論じようとする学問領域で，その歴史は浅く，まだやっと10年である．

幾何学図形の処理は，地理情報処理，VLSI(超高集積回路)，グラフィックスなどの多くの分野で必要とされるにもかかわらず，従来は，それぞれの分野でその場に応じた工夫を用いて行なわれていたようであるが，計算幾何学の進展によって次第に統一的な基本技術が確立される方向にある．もっとも，計算幾何学の研究の初期のものの多くは，計算機科学（computer science）の中の理論的興味に支えられた算法設計に終始していた感もあり，大規模な幾何学的データ処理の実用技術の確立を意識した研究が見られるようになったのは，ごく最近のことである．（つまり，最近ようやく理論的な枠組みがある程度整い，実用化を考える余裕が出てきたということであろう）

計算幾何学全般については，地理情報処理の立場からの本会報文集［4］もあり，また，最近読みやすい解説記事［2］も出ているので，それらを参照していただくとして，本稿では，1つのわかりやすい問題――平面マッチング問題――に焦点を絞ろう．そして，その近似解法の設計と解析を通じて，実際の大規模データを扱うさいの基本的手法であるバケット法［1］の有効性を例証しよう．

むろた　かずお　筑波大学　社会工学系
〒305　茨城県新治郡桜村

## 2. 平面マッチング問題とプロッター描線順序

### 2.1 平面マッチング問題

以下で考える平面マッチング問題というのは，平面上に$n$(偶数)個の点$P_i=(x_i, y_i)(i=1,\cdots,n)$が与えられたとき，2点ずつを組にして$n/2$個の対を作って，対になった2点間の距離の総和（マッチングのコスト）を最小にする問題である．ここで，2点$P_i, P_j$間の距離$d(P_i, P_j)$としては通常のEuclid距離($L_2$距離)：

$$d_2(P_i, P_j)=\sqrt{(x_i-x_j)^2+(y_i-y_j)^2}$$

を用いることが多いが，$L_\infty$距離：

$$d_\infty(P_i, P_j)=\max(|x_i-x_j|, |y_i-y_j|)$$

を使うこともある．

この問題は，一般のグラフ上で最小重み最大マッチングを求める問題の特殊な場合である．すなわち，$P_i(i=1,\cdots,n)$を頂点とする完全グラフ（すべての頂点対のあいだに無向枝をもつグラフ）において，枝$P_iP_j$に両端点間の距離$d(P_i, P_j)$が重みとして与えられていると考え，対になった2点$\{P_i, P_j\}$をマッチングの枝に対応させればよ

い．

一方，一般のグラフにおいて，その枝に重みが与えられたときの最小重み最大マッチング問題に対しては，グラフの頂点数の3乗程度の計算量で厳密な最小重みマッチングを与える組合せ論的な算法が知られている(くわしくは［6］,［7］などを参照のこと)．

したがって，この算法を平面マッチング問題に適用すれば $O(n^3)$ の手間で最適解が求められることになる．つまり，組合せ最適化の分野の通常の価値観によれば，平面マッチング問題は“解けた”わけである．

それにもかかわらず近似解法を考えようとするのはそれなりの実用上の要請があるからである．

### 2.2 プロッター描線順序の最適化

道路網のような図をXYプロッターで描こうとするときに，多数の線分をどのような順序でかくのが良いか，という問題は，プロッターの横で待ちつづける当事者にとっては重要問題である．というのは，プロッターのペンは1本の線分を描き終えると次の線分のところまでペン・アップの状態で動いていかなければならないから，でたらめな順序を指定されたペンは大方の時間を空中移動に浪費することになってしまうからである．

描くべき図形が直線分を枝とする連結な無向グラフであるとしよう．ペンが1本の枝を描き終ったとき，その終点がまだ描いていない別の枝の端点になっていれば，ペンの空送りなしに新しい枝へと描き進むことができる．グラフ理論の用語では，ある点に集まっている枝の本数をその点の次数というが，このような“乗継ぎ”がいつでもうまくいくのは偶数次数の点である．奇数次数の点では最後に乗継ぎ先がなくなってしまうので，1本の枝は2度描かないとすれば，どうしてもペンを上げて別の点に移動しなければならない．その行き先の点はどこでも良いのであるが，空送りの総回数を最小限に抑えるために，やはり奇数次数の点に選ぶ．このとき，空送り枝の本数は，奇数次数の点の数 $n$ の半分である（$n$ は必ず偶数であることに注意)．

ペンの空中移動総時間が空送り枝の総長に比例すると考えよう（XYプロッターの動作原理を考えると，空送り枝の長さとしては両端点の $L_\infty$ 距離を用いるのが適当である)．後者ができるだけ小さくなるように空送り枝を定める問題は，奇数次数の点 $P_i(i=1,\cdots,n)$ に対する平面マッチング問題に他ならない．

すなわち，この平面マッチング問題を解いて，対になった2点間を空中移動し，それ以外は“乗継ぎ”をしながらペン・ダウンで実質的な作図をつづけて行なうことによって最適描線順序が得られる．

ところで，大規模な道路網などでは，$n$ が数千から数万におよぶことも珍しくないので，描線順序を定めるためだけに $O(n^3)$ の手間をかけて平面マッチング問題の厳密な最適解を求めるというのは実際的でない．当面の目的のためには，奇数次数の点をすべて対にすることはぜひ必要であるが，そのマッチングのコストを厳密に最小化しなくても空送り長が多少ふえるだけであり，あまり困らない．

すなわち，最小化のほうは多少妥協しても，より少ない手間でマッチングを作り出す近似算法を用いるほうが実際的観点からは重要である――というのが，平面マッチング問題の近似解法を考えようとする1つの動機である．

プロッターの描画図形が非連結である場合や枝の2度書きを許す場合を含めた描線順序の最適化については，実験例とともに，［2］,［5］,［8］にくわしい報告がある．

## 3. 近似算法

平面マッチング問題の近似解法をいくつか紹介する．この問題では目的関数が人間にとってわかりやすいものであるから，直観的にもっともらしい算法が良い算法であることが期待される．

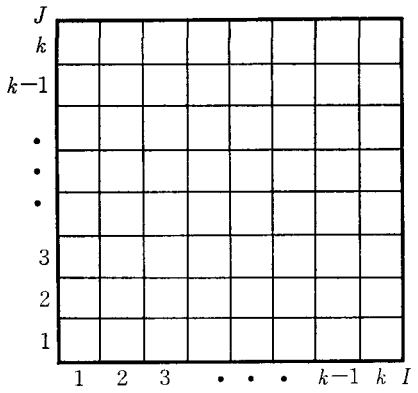



図 1 バケット分割

まず思いつくのは，最も近い2点から順番に対にしていくという貪欲算法であろう．素朴に，全点対間の距離を計算して，$n(n-1)/2$ 個の数字を小さい順に並べてやるとすれば，算法はたしかに単純である．が，全点対間距離の計算だけでも $O(n^2)$ の手間がかかり高速とは言いがたい．

ちょっと反省してみると，この貪欲算法は平面マッチング問題の幾何学的性質を何も利用していないことに気づく．さらに進んで，点間距離を表わす $n(n-1)/2$ 個の数字は，点の位置を表わす $2n$ 個の数字 $(x_i, y_i)$ $(i=1, \cdots, n)$ によって定まっているので互いに独立ではないという特殊性を利用すべきだと気づく．

以下，$n$ 個の点 $P_i(i=1, \cdots, n)$ は単位正方形 $S=\{(x,y)\,|\,0<x<1, 0<y<1\}$ の中にあるとする．点の分布に極端なかたよりがないとすれば，全領域 $S$ をいくつかの小領域に分けて考えればよさそうである．

最も簡単な分割として，図1のように $k^2$ 個の小正方形（バケット）に分けて，バケットに $(I, J)$ $(1\leq I\leq k, 1\leq J\leq k)$ という番地をつける．$k$ は，後に定めるパラメタ $\alpha$ を用いて $k\fallingdotseq\alpha\sqrt{n}$ とするが，このとき1つのバケットには平均 $1/\alpha^2$ 個の点が含まれることになる．点 $P_i=(x_i, y_i)$ が属するバケットの番地は，$I=[kx_i]+1$, $J=[ky_i]+1$ ([ ] は Gauss 記号)で計算できるので，全点をバケットに振り分ける手間は $O(n)$ で済む．

1つのバケットに属する点同士は“近く”にあるのであるから，バケット内では任意に対を作ってしまうことにしよう．すると，奇数個の点を含むバケットでは1点ずつの“残り”が出る．これをどう処理するかについては，少なくとも2つの方法が思い浮かぶ．

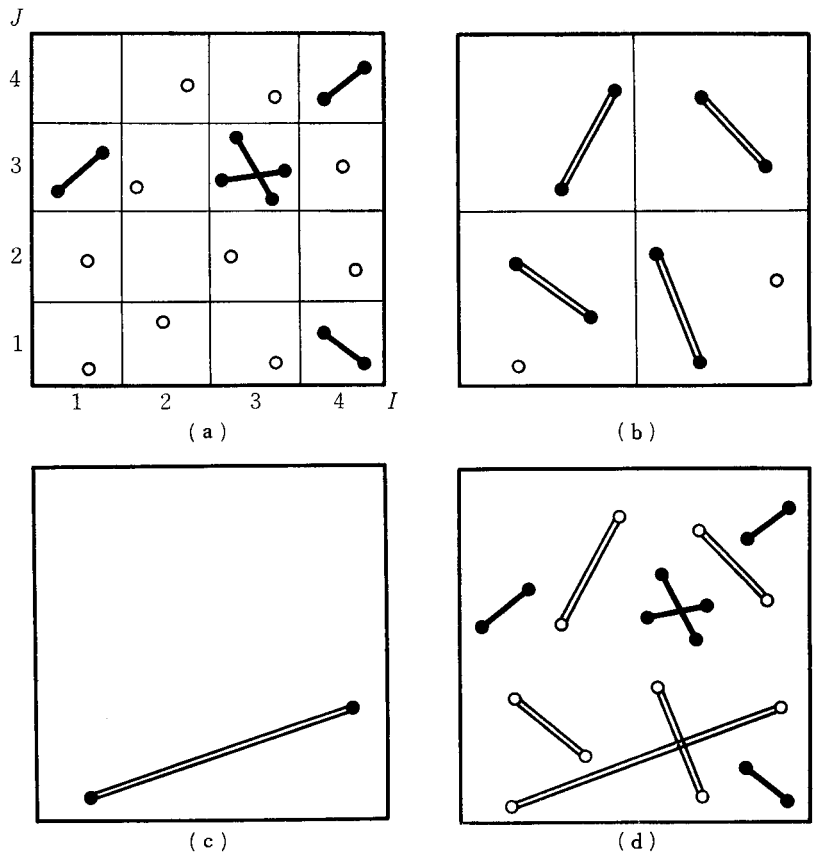
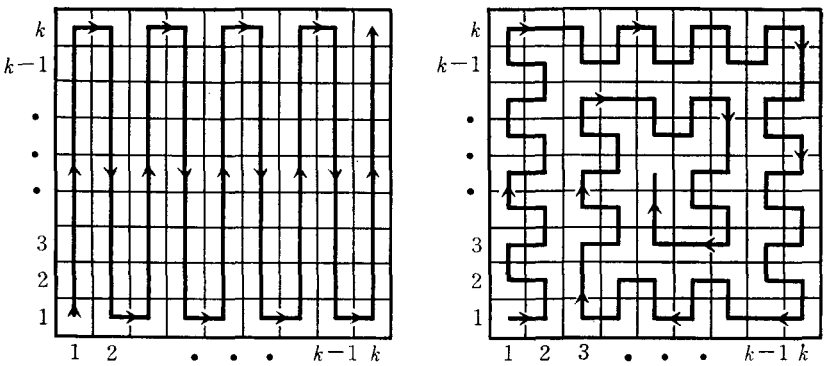



図 2 Bottom-up four-square 法 ($n=20$, $k=2^2$)
●：対になった点　○：残っている点

第1の方法は，近所のバケットを4つ集めて一段大きなバケットを考え，その中に2つ以上の点が残っていれば任意に対を作り，それでも残りがあればさらに大きなバケットを考えて同じことをくりかえすというものである（$k=2^m$ の形に選んでおく必要がある）．




(a) Serpentine 順　　(b) Spiral rack 順

**図 3** バケットをめぐる順番

この算法は[5]で bottom-up four-square 法（BUFS 法と略す）と名づけられている方法である．図2(a)～(d)に例を示す．

まず最初に5組の対(•—•)がバケット内にできて，10個の点(◦)が残る(図(a))．次に2組の対ができて2点が残り(図(b))，最後に1組の対ができて(図(c))終了する(図(d))．この算法は $O(n)$ の手間で実行できる．

第2の方法は，あらかじめ定めた順番にしたがって $k^2$ 個のバケットをめぐり，残っている点を拾っては対にしていく方法であり，[5]では前処理つき straightforward 法(SP 法と略す)と呼ばれている．バケットをめぐる順番にはいろいろ考えられるが，図3の(a)serpentine 順や (b)spiral rack 順などがある．図2(a)の例で serpentine 順を採用すると，○印の10点は図4のような順番で拾われて，5組のマッチング(◦═◦)が作られる．

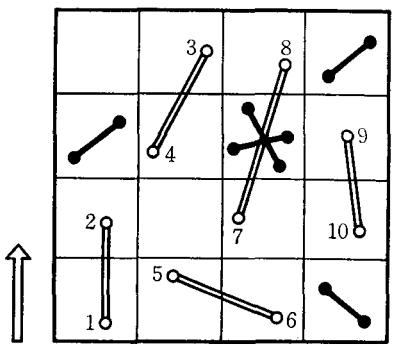



**図 4** Straightforward 法（前処理付，serpentine 順）

バケットの個数は $k^2=O(n)$ であるから，SP法もまた $O(n)$ の手間で実現できる．

SP 法においては，あらかじめ定められたバケット順にしたがって“残り”の点(◦)に $1,2,\cdots,\tilde{n}$ と番号をつけ，$\{1,2\},\{3,4\},\cdots,\{\tilde{n}-1,\tilde{n}\}$ のように対を作ったわけであるが，この組合せ方を1つずらして，$\{2,3\},\{4,5\},\cdots\{\tilde{n}-2,\tilde{n}-1\},\{\tilde{n},1\}$ としたものも同時に考え，両者のうちコストの小さいほうを採用することにすれば，マッチングのコストを小さくできる．この算法は，巡回方式の前処理つき straightforward 法（略して SPT 法）と呼ばれる[5]．距離計算のために手間は増えるが，やはり $O(n)$ の計算量で済む．

このように次から次へといろいろな着想を並べていくと，いくらでも“もっともらしい算法”が出てきそうである[3]．上に述べた BUFS 法，SP 法，SPT 法はいずれも $O(n)$ の手間で実現できるが，実際の計算時間(CPU 時間)と得られるマッチングの質(コスト)を天秤にかけるとき，どれを採るのが得か．どれかの方法に決めたとしても，バケットの大きさを定めるパラメタ $\alpha(=k/\sqrt{n})$ やバケット順などはどうすればよいのか，…．あまりに自由度がありすぎて，かえって収拾がつかない．直観に期待する楽観主義算法論も怪しくなってきてしまった．仕方がないので，いくぶん定量的に諸算法の性能を見積ってみよう．

## 4. 算法の性能

### 4.1 厳密解のコスト

そもそも，真の最小マッチングのコストはどの程度なのであろうか．本節でも $n$ 個の点が単位正方形 $S$ 内にあると仮定する．

最小マッチングの $L_2$-コスト $M_{OPT}(n)$ は $n$ 点の配置によって変わるが，$n$ 点が $S$ 内に一様分布するとしてその期待値 $E[M_{OPT}(n)]$ を考えよう．$n$ が十分大きいとき，ある1点から最も近い別の点までの距離は $O(1/\sqrt{n})$ である．最小マッチングは最も近い点同士を対にしたものではないけれど，最小マッチングで対になる点間の距離も $O(1/\sqrt{n})$ にちがいないから，

$$E[M_{OPT}(n)]=(n/2)\times O(1/\sqrt{n})=O(\sqrt{n})$$

と見積れる．実は，$M_{OPT}(n)/\sqrt{n}$ がある定数 $\beta$ に収束することが数学的に証明されており，その値は $\beta\fallingdotseq 0.32$ であることが実験的に［5］［9］知られている：

$$E[M_{OPP}(n)]\sim\beta\sqrt{n},\quad \beta\fallingdotseq 0.32 \tag{4.1}$$

点の配置が一様分布からかけ離れている場合が心配なので，$n$ 点のあらゆる配置に対する $M_{OPT}(n)$ の上限値 $\hat{M}_{OPT}(n)$ もおさえておこう．たとえば，$\hat{M}_{OPT}(2)$ は $S$ 内の2点間距離の最大値であるから $\hat{M}_{OPT}(2)=\sqrt{2}$（$L_2$ 距離）である．$n$ が大きいとき，

$$0.537\sqrt{n}\leq\hat{M}_{OPT}(n)\leq 0.707\sqrt{n} \tag{4.2}$$

となることがわかっている．

### 4.2 近似解のコスト

さて，本題の近似算法の性能解析にはいる．SP法を例に取ろう．上と同様にして，$M_{SP}(n;k)$ は（ある特定のバケット順を定めた）SP法（$k$ は単位正方形 $S$ の1辺あたりの分割数）によって作り出されるマッチングのコストを表わすものとし，$n$ 個のあらゆる配置に対する $M_{SP}(n;k)$ の上限値を $\hat{M}_{SP}(n;k)$ とかく．$\hat{M}_{SP}$ の評価が当面の課題である．

SP法で特定のバケット順を定めたとき，2点 $P_i$, $P_j$ が含まれているバケットの番号の差を，$P_i$ と $P_j$ の**バケット距離**と呼ぶことにする．同じバケットに属する2点のバケット距離は0である．

SP法によって作り出される対のうち，対のバケット距離が $j-1$ に等しいものの個数を $n_j$ とかく（$j=1,2,\cdots,k^2$）．$n_1$ はSP法の第1段で各バケット内で作られる対の数に等しい．対の総数は $n/2$ であるから，

$$\sum_{j=1} n_j=n/2 \tag{4.3}$$

が成り立つ．図4の例では，

$$n=20,\ n_1=5,\ n_2=4,\ n_3=1,\ n_4=n_5=\cdots=0$$

である．

さらに，$n_2+n_3+\cdots$ は，第1段で“残った”点同士から成る対（図4の(○=○)）の数である．バケットを巡回しながらこれらの対を作っていくとき，バケット距離が $j-1$（$j\geq 2$）である対を1組作るためには，バケットを $j$ 個分前進しなければならない．ところが，バケットの総数は $k^2$ であり，しかも1つのバケットには高々1点しか“残り”の点がないから，

$$\sum_{j=2} j n_j\leq k^2 \tag{4.4}$$

という関係式が成り立つ．

一方，バケット距離が $j$ である2点間の $L_2$-（あるいは $L_\infty$-）距離の上界値を $c_j$（$j=1,2,\cdots$）とかくことにすると，明らかに

$$M_{SP}(n;k)\leq\sum_{j=1} c_j n_j \tag{4.5}$$

である．$c_j$ の値は，バケットをめぐる順番と $k$ によって定まるが，たとえば，serpentine 順のときは，

$$\begin{aligned} &L_2\text{ 距離}: c_j=\sqrt{1+j^2}/k\quad (j=1,2,\cdots)\\ &L_\infty\text{ 距離}: c_j=j/k\qquad (j=1,2,\cdots)\end{aligned} \tag{4.6}$$

とおくことができる．

今や，次のLP問題を考えるのは自然である（$n_j$ は整数とは限らない）：

$$\boxed{\begin{aligned} &f\equiv\sum_{j=1} c_j n_j\to\max\\ &\text{s.t.}\quad (4.3),\ (4.4),\ n_j\geq 0\end{aligned}} \tag{4.7}$$

(4.5) により，このLP問題の最適値 $\hat{f}$ は $\hat{M}_{SP}$ の上界を与える．

このようにして，$\hat{M}_{SP}$ の上からの評価が(4.7)のLPを解くことによって得られることがわかったが，

(4.7)は変数の数が多くて大変なので，その双対問題：

$$
\begin{aligned}
& g \equiv \frac{n}{2} x + k^2 y \rightarrow \min \\
\text{s.t.} \quad & x \geq c_1 \\
& x + jy \geq c_j \quad (j = 2, 3, \cdots k^2) \\
& y \geq 0
\end{aligned} \tag{4.8}
$$

を考える．LPの双対性により，(4.8)の実行可能解に対する $g$ の値は $\hat{f}$ の上界であり，特に最適値 $\hat{g}$ は $\hat{f}$ に等しい．すなわち

$$\hat{M}_{SP}(n\,;k) \leq \hat{f} = \hat{g} \leq g. \tag{4.9}$$

という具合に，$\hat{M}_{SP}$ の評価は2変数のLP問題に帰着された．実際に(4.6)を代入して計算すると，serpentine 順を採用したSP法について，$n \rightarrow \infty (k \fallingdotseq \alpha\sqrt{n})$ のときに

$$\frac{\hat{M}_{SP}(n\,;\alpha\sqrt{n})}{\sqrt{n}} \lesssim \begin{cases} 1/(\sqrt{2}\alpha) + \alpha & (L_2\text{-距離}) \\ 1/(2\alpha) + \alpha & (L_\infty\text{-距離}) \end{cases} \tag{4.10}$$

を得る（実は，これらの評価が漸近的に“＝”となることも容易にわかる）．

$L_2$-距離を用いるとして $\alpha = \alpha_0 \equiv 2^{-1/4}$ とおくと，

$$\hat{M}_{SP}(n\,;\alpha_0\sqrt{n}) \lesssim 2^{3/4}\sqrt{n} \fallingdotseq 1.682\sqrt{n} \tag{4.11}$$

となるが，これは厳密解のコスト(4.2)の3倍程度である．手間をかけずに済ませたのだから，十分満足すべき性能である．(なお，誤解のなきよう書き添えるが，任意の点配置にSP法を適用したときのコストが，その配置に対する最小マッチングのコストの3倍以内だと保証しているのではない)

SP法でバケット順を変えたときには，(4.6)の $c_j$ の値をかえれば同じ解析ができる．SPT法についても同様にして解析ができるが，くわしくは[5]にゆずる．

第3節に述べた近似算法の中で，コストの上限値が最も小さいのはspiral rack順を用いたSPT法であり，$n$ 点に対する上限値は $0.9\sqrt{n} \sim 1.0\sqrt{n}$ 程度であることがわかっている．すなわち，この算法によれば，$O(n)$ の手間しかかけずに，厳密解のコストの2倍以下にできるわけであるから，プロッターの描線順序決定への応用には十分ということになろう．

以上，コストの上限値ばかり考えてきたが，やはり，平均的な性能が気にかかる．最後に，$n$ 点が $S$ 内に一様分布しているときにSP法による解がどの程度になるかを見積ろう．以下の議論は，確率論の素養を疑われるほどに大雑把ではあるが，その結論は実験的に確認されているものである．

前と同様に，バケット距離が $j-1$ である対の個数を $n_j$ とし，そのような対のあいだの平均 $L_2$-(あるいは $L_\infty$-) 距離を（2点がそれぞれのバケット内で一様分布するとして）$\bar{c}_j$ とかくと，

$$E[M_{SP}(n\,;k)] = \sum_{j=1} \bar{c}_j E[n_j] \tag{4.12}$$

としてしまっても大過はない．$\bar{c}_j$ の値はバケット順と $k$ とによって定まり，初等的に計算できる．

十分多くの点が $S$ 内に一様分布するとき，大きさ $1/k^2 = 1/(\alpha^2 n)$ の1つのバケットが $j$ 個の点を含む確率 $p_j$ は，ほぼポアソン分布：

$$p_j \sim \frac{1}{j!}\frac{1}{\alpha^{2j}} e^{-1/\alpha^2} \quad (j = 0, 1, \cdots) \tag{4.13}$$

にしたがう．

SP法の第1段で“残り”となる点数 $n - 2n_1$ は奇数個の点を含むバケットの数に等しいので，

$$E[n_1] = (n - m_0)/2, \tag{4.14}$$

ただし，

$$m_0 \equiv n\sum_{j=1}^{\infty} p_{2j-1} = \frac{k^2}{2}\left(1 - e^{-2/\alpha^2}\right)$$

となる．さらに，$j \geq 2$ ついては，

$$E[n_j] = \frac{m_0}{2} \cdot \frac{m_0}{k^2}\left(1 - \frac{m_0}{k^2}\right)^{j-2} \quad (j = 2, 3, \cdots) \tag{4.15}$$

と考えられるので，(4.14)，(4.15)を(4.12)に代入すれば平均的なコストが計算できる．その結果，serpentine 順のSP法によるマッチングの $L_2$-コ

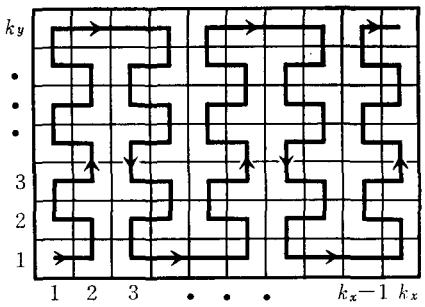



**図 5** Serpentine rack 順
($k_x$ は偶数, $k_y$ は奇数)

ストの期待値は $\alpha=0.79$ のとき最小となって $0.637\sqrt{n}$ 程度であることがわかる．これは厳密解の平均値 (4.1) の2倍であり，満足すべき値である．他の算法についても同様の見積りがなされている[5]．

## 5. おわりに

結局，どの近似算法を使えばよいのか．文献[5]では，理論的・実験的解析の結果を総合して，spiral rack 順の SPT 法($L_2$-距離のとき $\alpha=1.29$, $L_\infty$-距離のとき $\alpha=1.26$)を推奨している．このとき，マッチングのコストの上限は

$L_2$-距離：$1.04\sqrt{n}$ 以下，$L_\infty$-距離：$0.91\sqrt{n}$ 以下であり，平均値は

$L_2$-距離：$0.490\sqrt{n}$，$L_\infty$-距離：$0.449\sqrt{n}$

となる．

今まで正方形領域だけを考えてきたが，長方形の領域に対しては，$x$ 軸方向に長くなるように見て図5の serpentine rack 順の SPT 法を用いるとよい．

大規模な対象を扱うさいに部分領域(バケット)に分けて処理するというのはごく自然な発想であり，従来もいろいろな場面で用いられてきたにちがいないが，バケットを用いる算法のすべてが高性能だったというわけでもない．有用な算法を作るには，やはりそれなりの工夫が必要である．計算幾何学の諸問題へのバケット法によるアプローチが[1]にあるので，くわしくはそちらにゆずりたい．

### 参考文献


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

[2] 浅野孝夫，今井浩，伊理正夫：計算幾何学 1～5. *bit*, Vol. 16(1984), 1534-1541, 1647-1657; Vol. 17(1985), 108-116, 256-265, 368-376

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

[4] 伊理正夫，他：地理的情報の処理に関する基本アルゴリズム．日本オペレーションズ・リサーチ学会報文集 T-83-1, 1983

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

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

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

[8] 田口東：プロッターの描線順序の最適化．Proc. 3rd Math. Prog. Symp. Japan, 1982, 137-148

[9] Weber, M., and Liebling, Th. M.: Euclidean Matching Problems and the Metropolis Algorithm. Dept. of Math., Ecole Polytechnique Fédérale de Lausanne, 1985


**訂正とおわび**

前号12月号，特集記事「科学博に見るOR」(p. 750～753) におきまして，p.750に掲載いたしました福島総館長の御写真の下に，IBM館とはいっておりますが，これは，p.767の写真の下につくべきものでした．訂正して，おわびいたします．