設計自動化　40－19
(1987. 12. 18)

配線問題におけるタイル探索法の評価

Evaluation of a Tile-Search Method for Wire Routing

坂中　二郎・　小島　直仁・　佐藤　政生・・　大附　辰夫・
Jiro SAKANAKA　Naohito KOJIMA　Masao SATO　Tatsuo OHTSUKI

・早稲田大学理工学部
School of Science and Engineering, WASEDA University

・・拓殖大学工学部
Faculty of Engineering, TAKUSHOKU University

あらまし　レイアウト設計における配線手法として，迷路法のような格子を用いない，グリッドレスな配線手法が注目されてきている．タイル探索法では，配線領域をタイルと呼ばれる長方形に分割した図形情報として扱い，そのタイル上で探索を行うことによって，曲がり数最小の径路を高速に求めることができる．径路を求めた後に次の配線を行うために得られた配線径路を禁止領域とする必要があるが，その図形情報の更新は局所的な操作のみで行うことができる．今回，テストプログラムを作成することによってアルゴリズムの有効性を確認し，従来の線分探索法との比較もあわせてその実行速度等の性能評価を行った．

Abstract　Gridless routes find a simple-shaped path quickly and save memory space, because they do not construct a grid as in the maze-running methods, but rather treat routing problems as one of pattern processing in which the area to be used for interconnection wiring is taken as a figure itself. In this paper, we propose a new gridless router, named "tile-search method." It always finds a minimum bend path, if one exists, and updates the routing area so that the area for the path is considered to be unroutable. We also present performance data by experimental programs.

1. はじめに

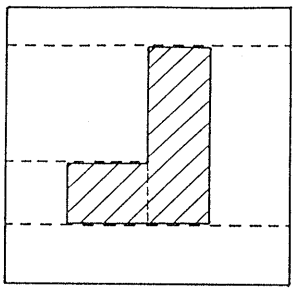
逐次配線処理における配線径路探索の手法として，従来，迷路法や線分探索法などがよく用いられている．迷路法[1]では配線領域の全格子点に対応するメモリが必要である．探索に要する時間も並列処理を行う専用ハードウェア等を用いない場合には，求まる径路の長さの２乗の時間が必要となる．

線分探索法[2,3]では，全格子に対応するメモリを必要とせず，径路探索の時間も求まる径路が単純な場合には高速である．ところが，多くの配線が引かれて配線領域の図形が複雑になってくると，その高速性は発揮されなくなり，あるいは手法によっては径路が存在しても発見できない場合があるなど，実用的とはいい難くなる．

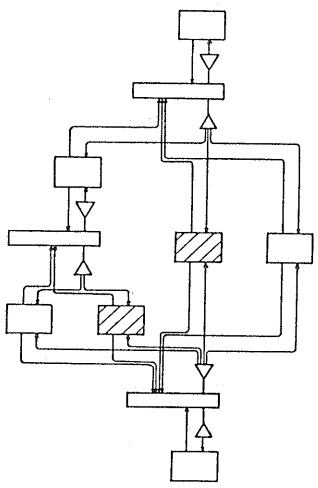
配線領域の長方形分割[4]と動的な線分探索問題を効率よく行うデータ構造[5]を用いると，O(n $\log^2$ n)の時間とO(n)の記憶容量，またはO(n log n)の時間と記憶容量で，曲がり最小の径路を求めることができる[6]（nは配線領域の図形としての頂点数）．また，この手法は格子に拘束されずグリッドレスな処理を行える．しかし，これに用いるデータ構造は，オーダーは保証されているものの結構複雑であって，係数的に不利である．また，１本の径路を求めた後に次の配線径路を求めるために得られた配線径路を禁止領域として図形情報を更新する必要があるが，その際にもこの複雑なデータの更新を行わなければならなかった．

そこで，線分探索のための複雑なデータ構造を用いずに最小曲がり径路を求める手法[7,8]を提案した．これにより，O(n log n)時間の前処理により，O(k log k)時間で径路を求め，O(k)時間で図形更新を行うことができる．ここでkは，O(n)個ある長方形のうち操作にかかわったものの数である．この手法を，長方形に分割された領域（これをタイルと呼ぶ）ごとに探索を行っていくことから，タイル探索法（Tile-Search Method）と名付けた．

ここでは，この手法に更に改良を加え，テストプログラムを作成してその性能を評価し，従来の線分探索法のテストプログラムとの比較も行ったので，その結果を報告する．



(a) 領域をタイルに分割



(b) タイルと分割線を結ぶポインタの構造

図１　タイル平面

2. 図形の長方形分割

タイル探索法では，図形をすべてタイルと呼ばれる長方形の集合として扱う．そのため，複合長方形をなす配線領域，禁止領域を長方形に分割する(図1(a))．メモリ量および実行時間のオーダーを保証するために，この分割は図形の凹な頂点を通り水平（垂直）方向に極大な分割線によって行われる．こうすることによって，長方形および分割線の数はO(n)に抑えられる．この処理は平面掃引算法によりO(n log n)の時間で実行できる．

なお，ここでは都合により，配線領域を水平に，禁止領域を垂直に分割しているが，必ずし

もその必要はなく，どちらも同方向に分割しても差し支えない．

得られたタイルの隣接関係を表現するために，分割線を介在として互いにポインタで指しあうデータ構造を用いる(図1(b))．長方形は上辺および下辺が接する分割線へのポインタを1つずつ持つ．逆に，分割線は自分の上下に接する長方形全てに対してポインタを持つ．このポインタを順次たどることによって，領域上の探索を行うことができるのである．しかしながら，ポインタで接続されているのは直接隣合うものどうしだけなので，図形の更新の際には局所的な更新だけで済む．

また，コーナースティッチング[3]のようなデータ構造に分割線の情報を加えて同様の機能を持たせることもできる．この場合には以下に述べる径路探索のアルゴリズムは少々異なるものとなるであろうが，基本的には同じ考え方で行える．

3. 分割線上のコスト

平面上の1点と始点Sを結ぶために必要な線分の数（即ち，最小曲がり数＋1）をその点のコストと呼ぶことにする．分割線上でのコスト分布は図2(a)のようになる．本アルゴリズムでは図2(b)（または(c)）のように，始点からの出発の方向を水平（または垂直）に限定して考えることにする．すると，水平に到達する部分のコストは奇数（偶数），垂直に到達する部分は偶数（奇数）となるが，これらをそれぞれ，水平コスト，垂直コストと呼ぶことにする．

水平な分割線上のある1点に他から垂直にコストnで到達できれば，分割線の残りの部分には少なくともコスト n＋1 で到達できることは明らかであろう．1本の分割線はその分割線から水平に到達できる点をすべて含んでいることから，分割線に他から径路が到達するのは垂直方向しかない．従って，分割線上のコストの分布は，

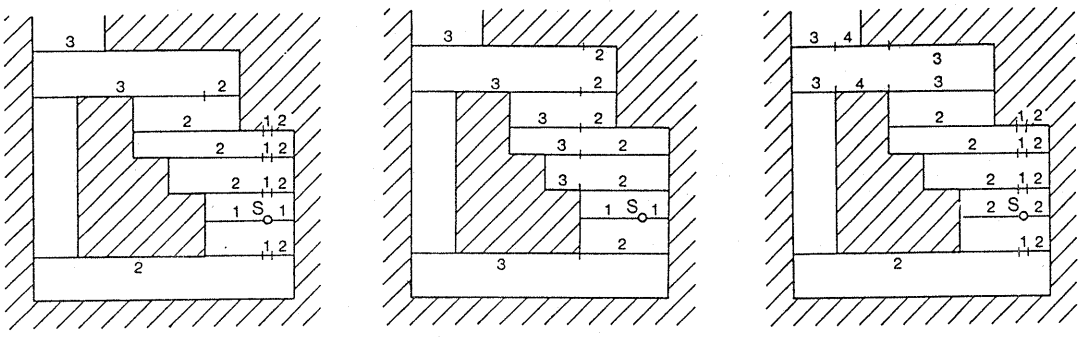
1. 垂直コスト n 一定
2. 垂直コスト n と水平コスト n＋1 が交互に分布

のいずれかのみであることがわかる．垂直コストは必ず存在するので，単に分割線のコストといえば垂直コストを指すものとする．

4. 径路の探索

ある分割線上のコスト分布がわかっていれば，そこからタイルを介して隣接する分割線に与えられるべきコスト分布は簡単な規則によって定まる．従って，2点間の最小曲がり径路を求めるためには，上に述べたコスト値を始点から始めて，隣接する分割線に順次与えていく．終点のコストが確定したらコストの伝達を終了し，逆に始点に向かって逆追跡を行って径路を定める．

コスト情報の伝達は，分割線のコストの小さいもの，さらにその中で伝達の進行方向の最も後ろにあるものから順に行われる．これは原理的にダイクストラ法と同じ手法であり，このよ



(a) 分割線上のコスト　(b) 水平に出発した場合　(c) 垂直に出発した場合

図2　分割線上のコスト

うにすることによって，重複したコスト伝達が行われず，コスト伝達の回数はO(k)に抑えられる．

また重要なのは，分割線が垂直／水平コストの分布を持つのは，現在伝達の行われている，迷路法でいえばウェーブフロントにあたる部分だけでよいという点である．隣接する分割線へのコスト情報の伝達を終わって，自身のコストが確定した分割線はコストの分布を持つ必要はなく，分割線について1つのコスト値を持つだけでよい．終点からの逆追跡により径路を決定する際にはそれだけで十分なのである．

そこで，分割線上のコスト分布を2-3木[10]で保持すれば，その根へのポインタを移す操作によりコスト情報の伝達を容易に行える．2-3木では，分割／併合が効率よく実行できるため，伝達先が複数に分かれる場合や，複数の径路から与えられるコストの有利な部分を結合させる処理も含めて，コスト情報の伝達は1回あたり，O(log k)時間で行える．また，探索の作業に要するメモリ量はO(k)となる．

従って，1本の径路を求めるのに必要な時間はO(k log k)となる．

5. 図形データの更新

配線径路が決定されたら，次の配線の際にその配線が禁止領域となるように，タイルと分割線の更新を行う．配線の各線分ごとに，配線幅に配線間隔を加えた長方形に覆われる部分を禁止領域に加える．この更新の作業は領域の長方形分割の際の条件を保つように行われる．

タイルの保持にコーナースティッチングを用いる場合には，Shand[10]の方法を用いることができる．ここで用いているデータ構造では，この手法を応用することによってやはり同様に図形の更新を行うことができる．

下にそのアルゴリズムを，また図3にこのアルゴリズムによって領域が更新される様子を示す．

この処理において，全ての操作は局所的に行われ，処理時間は操作にかかわるタイルの数に比例したものとなる．

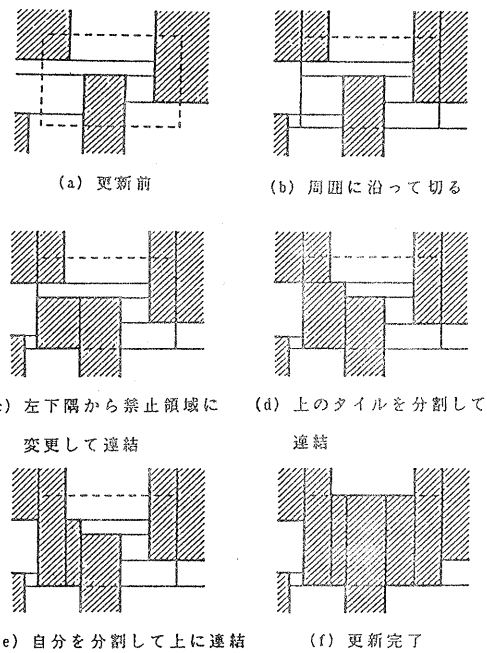


図3 領域の更新

```
領域更新アルゴリズム（x0, y0, x1, y1: 更新領域の座標）
begin
    更新する図形の右下隅を含む長方形に着目
    配線長方形ならばy0から下を切り離す
    着目長方形が右上隅に達しない間繰り返し
        着目長方形のx1から右を切り離す
        x1において上に隣接する長方形に着目
    end
    右上隅の長方形が配線長方形ならば，y1から上を切り離す
    更新領域の右辺に沿って並ぶ長方形について
        もし上下に隣接するものどうしの種別と右側のx座標が等しければ
            連結して1つの長方形とする
        end
    end
    更新領域の左側についてここまでと同様の処理
    更新領域内の左下隅の長方形に着目
    右端に達するまで繰り返し
        配線長方形ならばy0から下を切り離す
        種別を禁止長方形に変える
        着目長方形の上端がy1に達するまで繰り返し
            左端で上に隣接する長方形が配線長方形ならばy1より上を切り離す
            その右端が着目長方形の右端より右ならば
                その長方形をx1で切り離す
            そうでなくて，着目長方形の右端の方が右にでているなら
                着目長方形を上の長方形の右端のx座標で切り離す
            そうでもなければ
                着目長方形と上の長方形は既に同じx座標を持っている．
            end
            着目長方形に上の長方形を併合する
        end
    もし左隣の長方形が禁止長方形で上下端のy座標が等しければ併合する
    現在の着目長方形の下端で右に隣接する長方形に着目
    end
end.
```

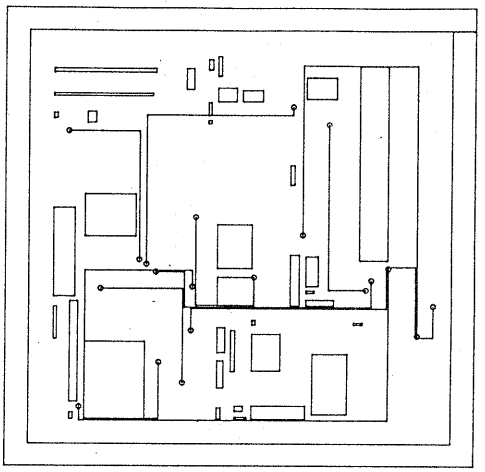


図４　配線の実行例

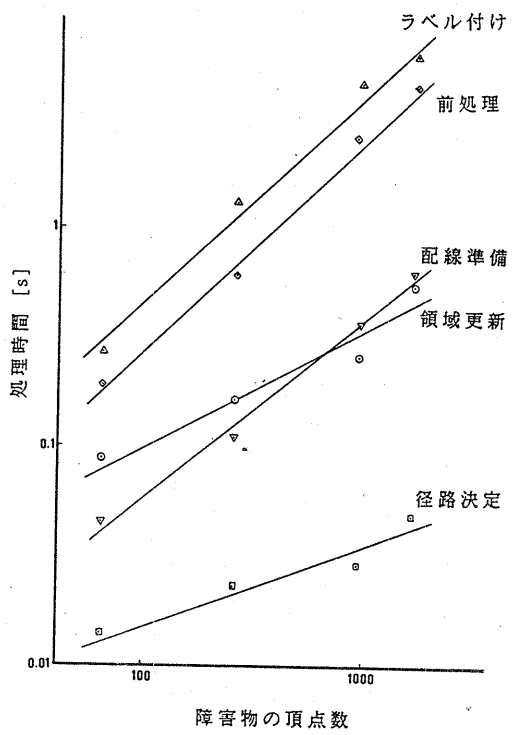


図５　障害物の頂点数に対する各処理時間

6．実験結果

本アルゴリズムの有効性を確認するために，実験用のテストプログラムを作成し，その実行時間等の性能の評価を行った．また，同時に単純な線分探索法による結果との比較も行った．

実験はＶＡＸ-11/785のＵＮＩＸ上で行われ，プログラムはＣ言語で記述されている．テストデータとしては，禁止領域を乱数により発生させた長方形の障害物とし，結線要求としては，やはり乱数によって配線領域内に設定した端子対を用いた．配線の実行例を図４に示す．

まず，障害物の頂点数nを変化させて，端子は全領域に均等に分布させた場合の各処理の実行時間を表１に示す．前処理以外の実行時間は連続して行った10本の配線の平均である．また，これを両対数グラフにプロットして図５に示す．これを見ると，前処理（領域の長方形分割），配線準備（ラベルのクリアと始点，終点の分割線の設定），ラベル付けがほぼ傾き１である．理論的に前処理はＯ(n log n)，準備はＯ(n)であり，必ず図形全体をひととおり処理するのでほぼその通りの値となっている．ただし，実験データでは，図形の形状によってばらつきができるため，正確にＯ(n)かＯ(n log n)かまでは判別し難い．

また，ラベル付けの時間は探索の行われた分割線の数kで効いてきて，Ｏ(k log k)であるが，この例では配線経路が図形全域にまたがっているので，ラベル付けはほぼ全体に拡がり，kはn

表１：障害物の頂点数に対する各処理時間

頂点数	全処理時間	前処理	配線準備	ラベル付け	経路決定	領域更新
64	4.77	0.19	0.035	0.271	0.014	0.086
256	18.10	0.62	0.110	1.340	0.023	0.165
932	59.79	2.66	0.377	4.791	0.029	0.260
1668	84.69	4.74	0.630	6.390	0.038	0.555

表２：配線の性質と処理時間

総配線長	ラベル付け	探索線分数	領域更新	長方形処理数
111804	2.256	1871	0.079	398
180644	3.220	3141	0.128	710
261513	2.992	2458	0.131	686
396634	2.985	2612	0.170	883
880502	4.791	4729	0.260	1532

とほぼ同じオーダーになっているものと考えられる．

これに対して，経路決定（逆追跡）と領域更新は傾きがそれぞれ 0.3，0.5と，図形の頂点数に対してはかなり低いオーダーとなっている．これはもちろん，これらの処理が図形全域にわたらず，配線経路の周囲のみで行われるためである．配線が図形全域にまたがっているとして，２次元に分布する長方形領域に対して１次元の配線経路に交わるものはおおよそ$O(\sqrt{n})$となると考えられるが，領域更新はこれに合致する値となっている．経路決定では横方向にたどるのは１回で行えるため，さらにnの影響を受けにくくなっているものと思われる．

次に，配線の違いによってどのような結果が得られるかを調べるために，障害物を一定にしておいて，乱数によって端子を発生させる位置を限定し，得られる配線長を変化させてみた．表２にこのときにのラベル付けの時間と探索の行われた線分数，および配線後の領域更新の時間とその際に処理した長方形の数を示す．同様に10本の配線の平均である．

図６に示す通り，ラベル付けに要する時間は探索線分数に比例することがわかる．また図７では，図形更新の時間もやはり長方形の処理の行われた数に比例している．ここで，図８に配線長と探索線分数，長方形処理数の関係をみると，探索線分数は傾き約0.3，長方形処理数の傾きがほぼ0.7となっている．ただし，これらの関係は配線の形状に大きく左右されるので，かなりばらつきがあって必ずしも直線に乗っているとはいえない．しかし，いずれも配線の長さに関して１乗より低いオーダーに抑えられていることに注目しておきたい．

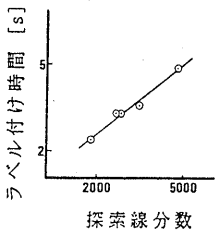


図６　探索線分数に対する探索時間

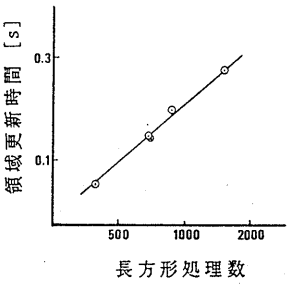


図７　長方形処理数に対する更新時間

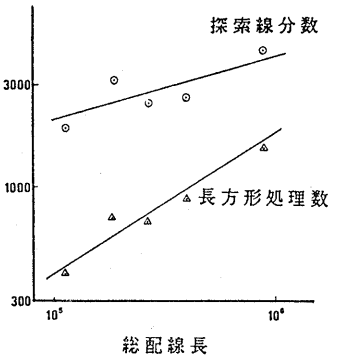


図８　配線の性質

また，従来用いられている線分探索法についてもテストプログラムを作成しこれと比較を行ってみた．線分探索法は，１つのレベルの線分上の各点から次のレベルの線分を発生させることによって，径路が存在すれば必ず発見するMiakmi-Tabuchiの方法によった．線分データは単純に線形リストで保持した．

障害物の頂点数に対する探索時間（連続10本の平均）を表３および図９に比較する．タイル探索法の方が高速に径路を求めていることがわかる．しかし，線分探索法の方が傾きが小さく，頂点数が増えるとタイル探索法の方が不利になりそうに見受けられる．これは，線分探索法では一定の格子数で障害物の数のみを変化させたためである．さらに頂点数の多い図形を収容するためには，格子を増やさねばならず，発生させる線分の数が多くなって実行時間は増大してしまうはずである．それに対し，タイル探索法では格子を用いないので，そのような制限はない．実際，この実験例では，タイル探索法では座標値を４バイトの整数で扱っているのに対し，線分探索法では格子は1000×1000としたから，扱うことのできる図形の複雑さに大きな違いがある．

得られた径路の曲がり数によって分類し，その径路を求めるために要した時間を調べて平均したものが，表４および図10である．線分探索法では１回，２回の曲がりの径路は非常に高速に求められるが，３回曲がり以上では急速に所要時間が増大している．これに対して，タイル探索法ではほぼ曲がり数に比例した時間で探索が行われている．

タイル探索法では，隣接する分割線にコスト情報を伝達していくだけであるから，探索が進

表３：障害物の頂点数に対する探索時間の比較

頂点数	線分探索法	タイル探索法
64	2.417	0.320
256	5.446	1.473
932	6.148	5.197
1668	10.828	7.058

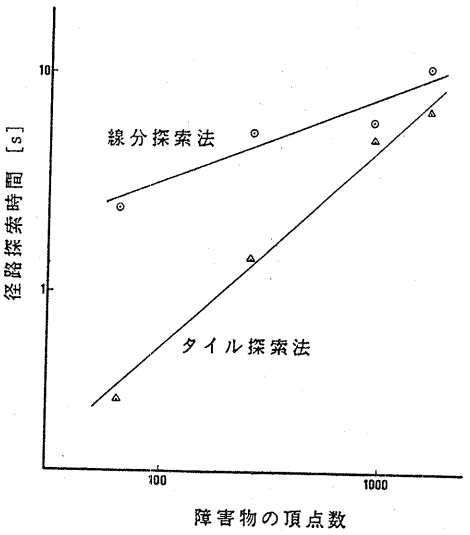


図９　障害物の頂点数による線分探索法との比較

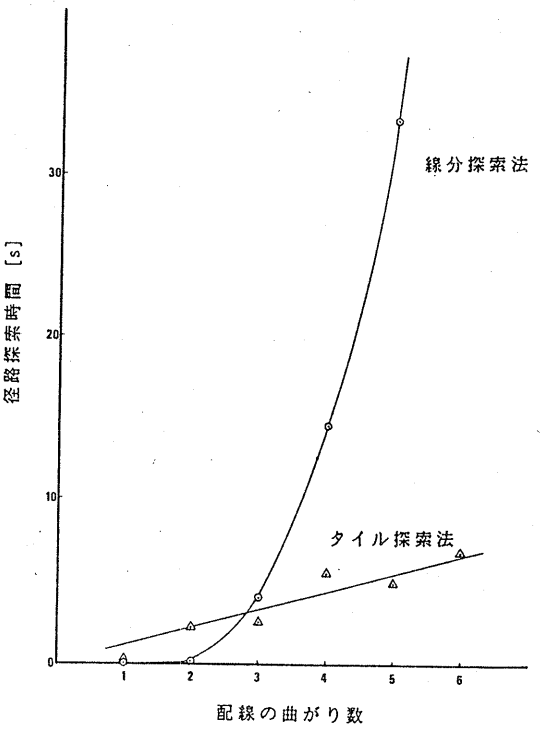


図１０　配線の曲がり数による線分探索法との比較

表4：配線の曲がり数に対する探索時間の比較

曲がり数	線分探索法	タイル探索法
1	0.059	0.243
2	0.259	2.223
3	4.096	2.553
4	14.612	5.568
5	33.425	4.943
6	—	6.805

むことによって探索対象が増えることはない．ところが，線分探索法では，最初は線分交差の探索対象が障害物だけであるのに対し，3回曲がり以上になるとそれまでに発生させた探索のための線分も探索対象となるため，探索が進むにつれてそれがどんどん増えていく．曲がり数が増えると事実上線分探索法では径路を求めることは不可能に近い．タイル探索法では曲がり数の多い径路でも妥当な時間で求められるものと思われる．

7. おわりに

本稿では，領域を長方形に分割した図形情報として扱い，その上で径路の探索を行うタイル探索法について述べ，プログラム実験の結果を報告した．実際の実行時間が，理論に合致していることを確認し，また従来の線分探索法と比較して複雑な径路の探索が高速に行えることがわかった．

参考文献

[1] C. Y. Lee, "An Algorithm for Path Connection and its Applications," IRE Trans. on Electric Computers, pp.346-365 (1961)

[2] K. Miakmi and K. Tabuchi, "A computer Program for Optimal Routing of Printed Circuit Conductors," IFIP Congress 68, pp. 1475-1478 (1968)

[3] D. W. Hightower, "A Solution to Line-Routing Probrem on the Continuous Plane," Proc. DA Workshop, pp. 1-24 (1969)

[4] 大附，佐藤，橘，鳥居，"複合長方形領域の最小分割，" 情処論文誌，Vol. 24, No. 5, pp. 647-653 (1983)

[5] 佐藤，大附，"動的な線分探索問題に対する算法とその評価" 信学技報，CAS85-86, pp.29-36 (1985)

[6] T. Ohtsuki, "Gridless Routers — New Wire Routing Algorithms Based on Computational Geometry," Internat. Conf. on Circuits and Systems, China (1985)

[7] 佐藤，坂中，大附，"線分探索法の高速化，" 信学技報，CAS85-154, pp. 49-55 (1986)

[8] M. Sato, J. Sakanaka and T. Ohtsuki, "A Fast Line-Search Method Based on A Tile Plane," Proc. IEEE ISCAS, pp. 588-591 (1987)

[9] J. K. Ousterhout, "Corner Stitching: A Data-Structuring Technique for VLSI Layout tools," IEEE Trans. CAD, vol. CAD-3, No. 1, pp. 87-100 (1984).

[10] A. V. Aho, J. E. Hopcroft and J. D. Ullman, "The Design and Analysys of Computer Algorithms," Addison-Wesley, Reading, Mass. (1974)

[11] M. A. Shand, "Algotithms for Corner Stitched Data-Structures," Algorithmica, No. 2, pp. 61-80 (1987)

盛光印刷所