設計自動化 48－6
（1989. 7. 11）

線分展開法の改良とその評価

小島直仁＋　　佐藤政生＋＋　　大附辰夫＋

＋ 早稲田大学理工学部
＋＋ 拓殖大学工学部

ＶＬＳＩやプリント基板の配線径路探索手法として迷路法と線分探索法が普及しているが，さらに効率が良く融通性の高い配線手法として，格子構造を用いずに図形処理により径路を求めるグリッドレス・ルータに関する研究がなされている．本稿では，グリッドレス・ルータの中でも線分展開法に焦点をあて，この算法を改良したアルゴリズムとレイアウト・データ管理システムを提案する．また，提案した算法をヒープ探索木を用いてインプリメントし計算機実験を行った結果を示す．

Improvement of the Line-Expansion Algorithm and its Evaluation

Naohito Kojima+　Masao Sato++　Tatsuo Ohtsuki++

+ School of Science and Engineering, Waseda University
3-4-1 Ohkubo, Shinjuku-ku, Tokyo 169, Japan
++ Faculty of Engineering, Takushoku University
815-1 Tatemachi, Hachioji-shi, Tokyo 193, Japan

Various gridless routing algorithms have been investigated in the past decade. Among them, the line-expansion algorithm seems to be most promising in its efficiency and flexibility. A new fast router, which is based on the line-expansion algorithm, is presented in this paper. It finds a path, if one exists, in $O(k \log^2 n)$ time with $O(n)$ memory space by means of priority search trees, where n is the number of vertices of routable regions and k is the number of searched vertices ($k \leq n$). Performance data of an experimental program is also presented.

1．まえがき

ＶＬＳＩやプリント基板における逐次配線手法としては，迷路法と線分探索法が最も普及している．しかし，これらは本質的に配線領域上に格子構造を想定しているため処理が単純である反面，多大な処理時間とメモリを必要とし，プロセス技術の進歩に伴う設計規則の変化に追従しにくいという問題点を持つ．これらの問題点を解決するための一つのアプローチとして，格子構造に依存しない配線手法（グリッドレス・ルータ）に関する研究がここ十年来行われている．

提案されているグリッドレス・ルータはその径路探索方法により以下の４つに分類できる．

① 可視グラフを作成し，これを用いて径路を求める方法[1]

② 配線領域を長方形に分割し，隣接する長方形を順に探索することにより径路を求める方法[2,3]

③ 補助線をあらかじめ発生し，交差する補助線を順に探索することにより径路を求める方法[4,5,6]

④ 配線領域を逐次探索することにより径路をもとめる方法[7]

これらのうち，①，②，③の方法は配線禁止領域や既配線といったレイアウト・データ以外にグラフや線分などの補助的なデータを必要とするため少なくともメモリの使用効率上問題がある．一方，④の方法は，

・図形データを直接扱うので複雑な設計規則に対応し易い，
・配線結果をマスク・パタンに変換し易い，
・迷路法の様に格子を用いないので使用メモリが少ない，
・径路が存在すれば必ずこれを求める，

といったグリッドレス・ルータの特徴の他に，

・配線領域全体に及ぶ前処理がない，
・メモリの使用効率が良い，
・個々の配線に対してそれぞれ独立した設計規則を用いることが容易である，

などの特徴を持ち，実用性の観点から見ると非常に望ましいグリッドレス・ルータとなる．④の方法としては線分展開法[7]があげられる．しかし，これは単純なデータ構造を用いているために$O(kn)$の手間を要してしまい実用的とは言い難い．ここで，ｎは配線禁止領域の頂点数，ｋは探索した頂点数である．

そこで本稿では，まず線分展開法を改良した算法を提案し，次に，この算法を支援するレイアウト・データ管理システムを提案する．最後に図形情報探索用データ構造にヒープ探索木を採用してインプリメントを行ったときの計算機実験結果を報告する．このときの時間複雑度は$O(k\log^2 n)$，空間複雑度は$O(n)$となる．

以下では議論を簡単にするために配線領域は垂直および水平な線分で囲まれた領域（複合長方形領域：rectilinear region）とする．また，一層で２点Ｓ，Ｔ間の配線を行うものとし，このとき求める配線径路は斜め線を含まないものとする．各配線はその中心線で表現されるが，配線幅や配線間隔を考慮しているので設計規則に違反することはない．

2．改良線分展開法

提案する改良線分展開法は基本的にはダイクストラ法に基づくものである．まず図１(b)に示すように始点Ｓを通る線分を発生させ，この線分を垂直方向に障害物にあたるまで掃引（展開）することによって配線領域を探索する．探索した長方形領域に終点Ｔがなければ，この長方形の外周上に必要に応じて線分を発生させ同様の領域探索を行う．探索を繰返してゆきＴに到達したならば逆追跡によって径路を求める（図１(g)参照）．以下にアルゴリズムの詳細を示す．ここでは，発生した線分のことを active line，active line を後述のコストに基づいて保持し次に展開させる active line を得るためのデータ構造を priority queue，active line を幾何的に線分として保持し高速に２次元線分探索を行うためのデータ構造を線分探索用データ構造と呼ぶ．また，配線禁止領域や端子，既配線を幾何的に保持し高速に２次元図形探索を行うためのデータ構造を図形探索用データ構造と呼ぶ．線分探索用データ構造と図形探索用データ構造は互いに独立なものになっている．それは前者がルータのみが使用するものであるのに対し，後者は３節で述べるように配置算法，レイアウト・スペーサなど他のアプリケーション・ソフトウェアも使用する共用データ構造で

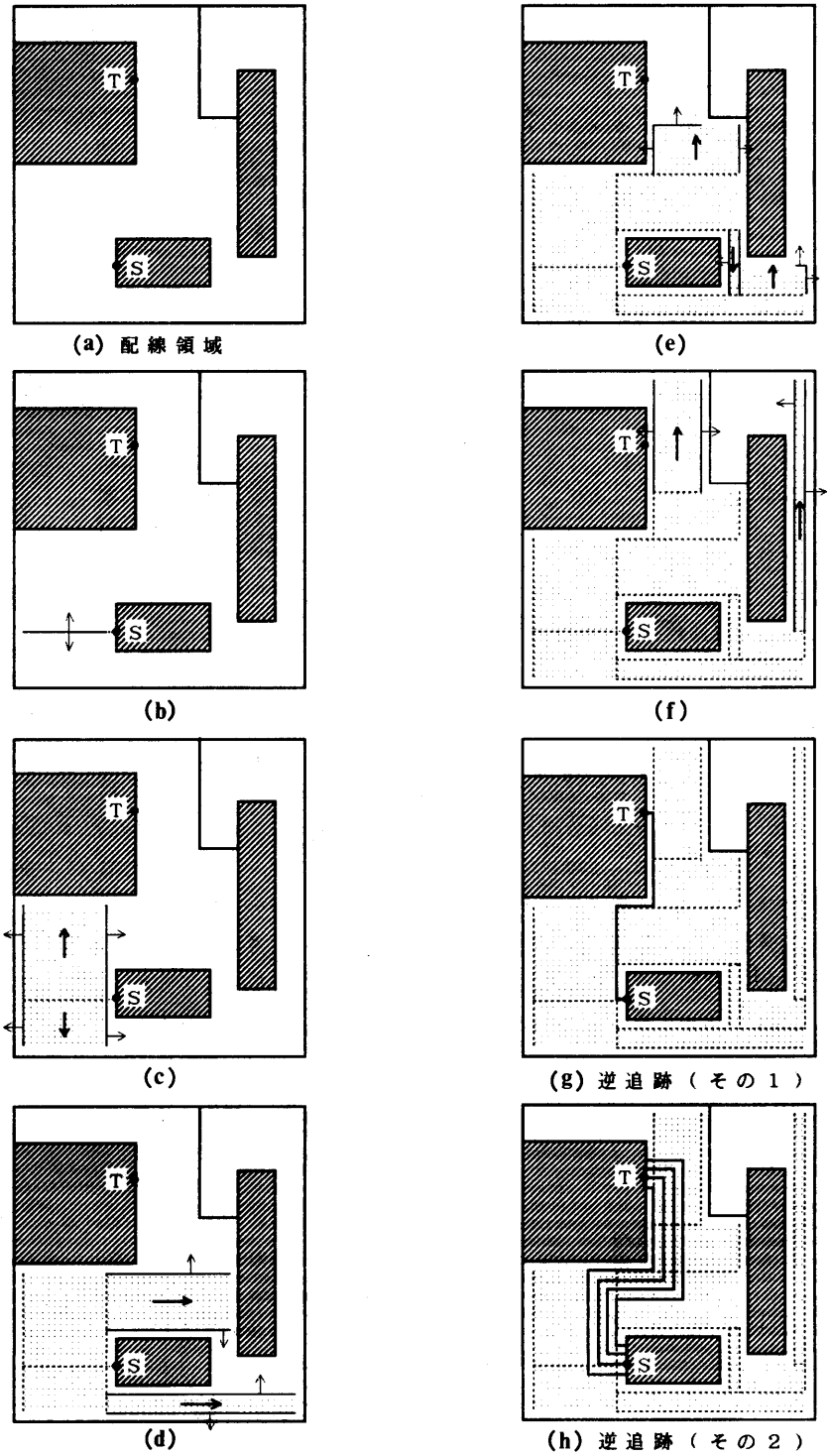


図１：改良線分展開法の径路探索手順

あるからである．

Step 0

priority queue，線分探索用データ構造の初期化を行う．これらは最初，空である．

Step 1

配線領域内にＳを通る線分を発生させる．発生させた線分上にＴが存在すれば，径路を得て終了．発生させた線分 active line を priority queue および線分探索用データ構造に挿入する．このとき，これらにはコスト０を与える．

Step 2

priority queue から最もコスト（次の Step で述べる）の小さい active line Ｌ をひとつ取り出す．priority queue が空なら径路は存在しない．この場合は Step 6 へ．

Step 3

Ｌから探索方向で最も近い図形もしくは他の active line を図形探索用データ構造と線分探索用データ構造を探索することにより求める．この操作はＬをＬとは垂直方向に障害物に衝突するまで掃引（以後，展開と呼ぶ）することにより長方形領域Ｒを得ることに相当する（図２参照）．このとき，既に存在する active line は線分探索の対象となるので，配線領域を active line が２重に展開することはない（図３参照）．Ｒ内にＴが存在する場合は Step 4 へ．

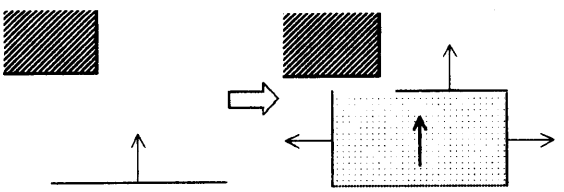


図２：active line の展開

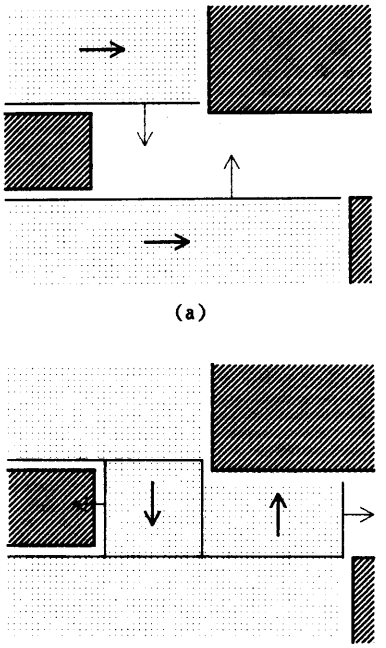


図３：active line の２重展開を防ぐ

Ｒの外周上に図２のように新しい active line を発生させる．ここで発生された active line にはすべてＬへのポインタを持たせ，コストを与える．ポインタは Step 4 の逆追跡の際に必要となる．コストは，種々のものが考えられるが，本稿では次式のように簡単な計算式の形で与えられるコストを採用した．

Ｇ＝ａＢ＋ｂＤ＋ｃＺ

Ｇ：Ｌに対するコストの増分

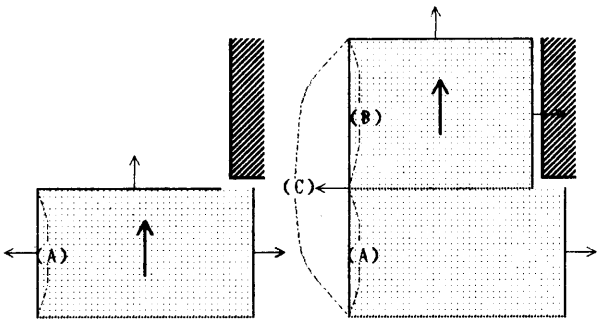
Ｂ：曲がりの有無（曲がりあり：１，曲がりなし：０）

Ｄ：Ｌからの距離

Ｚ：Ｔまでのマンハッタン距離

ここで，距離とは２つの線分の場合にはそれらの中点間の距離をさし，点と線分の場合にはその点と線分の中点間の距離をさすものとした．各パラメータの係数ａ，ｂ，ｃは各ネットについて独立に指定することができる重みである．ａの値を大きくすると曲がりの少ない径路が求まる傾向が強くなり，ｂの値を大きくすると短い径路が求まる傾向が強くなる．ｃはＴに近づく方向の探索を優先させるものである．

発生した active line を priority queue と線分探索用データ構造へ挿入する．ただし，active line が図４のように直線上で隣接する場合には１本に併合する．Step 2 へ．



２本の active line(A),(B) は
併合されて (C) になる
図４：active line の併合

Step 4

逆追跡を行い径路を得る．各 active line は自分の親の active line へのポインタを持っているので，逆追跡の際はそれを一方的に追跡して行くことで径路が得られる．

単に径路を求めるだけであれば，これで十分であるが，Step 3 で active line Ｌを展開する際にＬに対して垂直な active line Ｑ，Ｒをダブルリンクすることによって配線領域を「領域」として認識しながら逆追跡を行うことが可能となり，必ずしも active line と一致しない径路を求めることができる．また，こうすることによって逆追跡時に配線可能な幅を計算できるので，束配線（バス配線[8]）などへの応用が考えられる（図５，図１(h)参照）．

Step 5

径路を構成する線分を図形探索用データ構造に挿入する．

Step 6

線分探索用データ構造が使用しているメモリを解放し，終了．

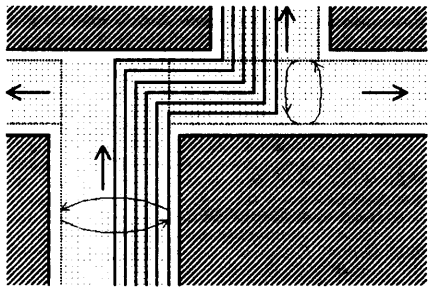


図５：向かい合う active line へのポインタ

３．レイアウト・データ管理システム：ＤＭＳ

現在，我々はグリッドレス・レイアウト・システムを構築中であり，本稿で提案するグリッドレス・ルータはこのシステムの中の１つのアプリケーション・プログラムとなっている．このルータの他には配置算法，スペーサ，ダイナミック・ルータなどが開発されつつある．各アプリケーション・プログラムは独立に動作するが，これらの間のデータのやりとりを効率良く行うために レイアウト・データ管理システム （Data Management System，以後ＤＭＳと呼ぶ）を開発した．ＤＭＳは各アプリケーション・プログラムが共用する一種のデータベースである．以下ではＤＭＳに関してさらに詳しく述べる．

3.1 ＤＭＳの意義

レイアウトシステムでは，ネットリストや設計規則等のデータファイルと設計者による指示（具体的にはキーボード，マウス等からの入力）が入力となり，これに各種配置配線処理アルゴリズムやユーザの操作が加わり，最終的に結線を行ったレイアウトパタンが出力される．レイアウト・データはある決められたデータ構造で格納されるが，複数のプログラムからアクセスされるためそれらに対する処理は独立したプログラムが一括して行うのがより効率的である．これによりレイアウト・データの統一化をはかることが可能となる．また，データに対する処理手順をデータ構造に依存しないものにすれば，ルータなどのプログラムがデータ構造から独立し，将来開発されるであろうより優れたデータ構造に置き換える際にはＤＭＳの一部を変

更するだけでアプリケーション・プログラムを変更することなしに対応することが可能となる．ＤＭＳはレイアウト・データの統一化，プログラムのデータ構造からの独立を実現するものである．

3.2 ＤＭＳのデータ構造

ＤＭＳが保持するデータ構造は大きく分けて以下に示す３つに分けられる．

(A) ライブラリデータ

部品（セル，素子など）の形状，また各部品に含まれるピンの情報，そしてピンやビアの形状，設計規則などのライブラリデータを持つ．

(B) ネットリスト

未配線ネットの場合には結線対象となる端子が，配線後のネットに対しては，配線径路を構成する線分，ビア，端子などのデータを持つ．

(C) 図形探索用データ構造

配線禁止領域，配線線分，端子，ビアのレイアウト情報を幾何的に探索可能な状態で構造化して保持している．配線領域データからは，端子と配線領域の境界が参照されている．

3.3 ＤＭＳに対する処理手順

ＤＭＳは自らの保持するレイアウト・データに対する処理手順を提供している．この手順は，ＤＭＳが内部で実際に用いている図形データ構造とは全く独立なものである．ＤＭＳのデータに対する処理手順は関数，マクロ関数で提供されており，ＤＭＳの保持するあらゆるデータに対してアクセスが可能である．これによってプログラマには直接ＤＭＳのデータ構造を意識することなくルータなどのプログラムを開発する環境が提供されることになる．

4．計算複雑度

改良線分展開法とＤＭＳの計算複雑度を，線分探索用データ構造と 図形探索用データ構造に ヒープ探索木[9]を用いた場合について考察する．以下で，ｎは配線禁止領域の頂点数，ｋは探索した頂点数である．ここでは扱うべき座標値の最大値がｎと同程度であるとして議論を行う．

ＤＭＳの保持するレイアウトデータは$O(n \log n)$の手間，$O(n)$のメモリで作成できる．ＤＭＳのデータに対する１回の図形探索は$O(\log^2 n)$の手間で実行される．ただし厳密にいえば，これは active line の長さが短く探索の幅が配線領域と比較して小さい場合であって，これが配線領域に対して充分大きい場合は$O(n)$となる．しかしこのようなことは極めてまれである．

改良線分展開法で，１回の線分の展開に要する手間は$O(\log^2 n + \log^2 k)$となる．展開はｋ回行われるので全体としては$O(k \log^2 n)$となる．他の処理は高々$O(k)$の手間で行えるので，改良線分展開法の時間複雑度は$O(k \log^2 n)$となる．また，空間複雑度は$O(k)$である．

以上の議論より，ＤＭＳによりレイアウト・データが作成された後は$O(k \log^2 n)$の手間で径路が求まることがわかる．

5．計算機実験結果

提案した改良線分展開法とＤＭＳをＣ言語を用いてインプリメントし，計算機実験を行った． 線分探索用データ構造と図形探索用データ構造にはヒープ探索木[9]を用いた．使用した計算機はＳＵＮ４／１１０である．配線禁止領域，配線要求等は乱数によって発生させた．比較のために改良線分探索法[6]に関するデータを合わせて報告する．

比較項目は，処理速度と使用メモリの量である．処理時間は，配線禁止領域の頂点数を変えそれぞれ１本ずつ１０回配線した結果の平均をとった．使用メモリについては，双方が使用するＤＭＳのメモリ量と，各手法が径路探索のために使用する独自のデータ構造のためのメモリ量を処理時間と同様の方法で測定した．処理時間の測定結果を表１と図６に，使用メモリの量を表２と図７に示す．

測定結果をみると，改良線分展開法の配線処理時間は概ね配線禁止領域の頂点数に比例する程度の値となっている．これは改良線分探索法の処理時間と比較してよいとは言えないが，改良線分探索法が前処理とし

表１：配線禁止領域の頂点数と処理時間の関係

配線領域の頂点数	改良線分探索法		改良線分展開法
	前処理 (Sec.)	配線処理 (Sec.)	配線処理 (Sec.)
72	0.06	0.10	0.07
104	0.08	0.11	0.10
136	0.13	0.14	0.24
200	0.17	0.17	0.21
264	0.25	0.28	0.34
392	0.38	0.35	0.50
520	0.59	0.47	0.68
776	0.87	0.53	0.80
1032	1.41	0.82	1.06
1544	1.96	1.30	3.50
2056	3.18	1.45	3.35
3080	4.43	2.54	6.11

表２：配線禁止領域の頂点数と使用メモリの関係

配線領域の頂点数	DMS (bytes)	改良線分探索法 (bytes)	改良線分展開法 (bytes)
72	6624	21864	7827
104	9598	25538	11689
136	12512	39403	20985
200	18400	47423	19910
264	24288	73110	33143
392	36064	89609	46444
520	47840	140238	59554
776	71392	169796	64845
1032	94944	269141	85476
1544	142048	339848	244970
2056	189152	528390	223632
3080	283360	657927	395733

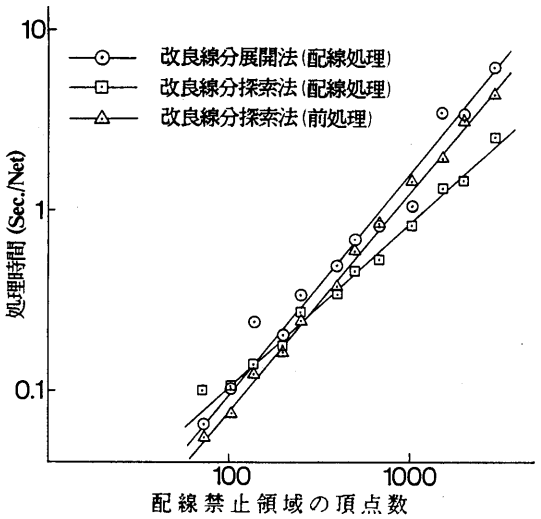


図６：配線禁止領域の頂点数と処理時間の関係

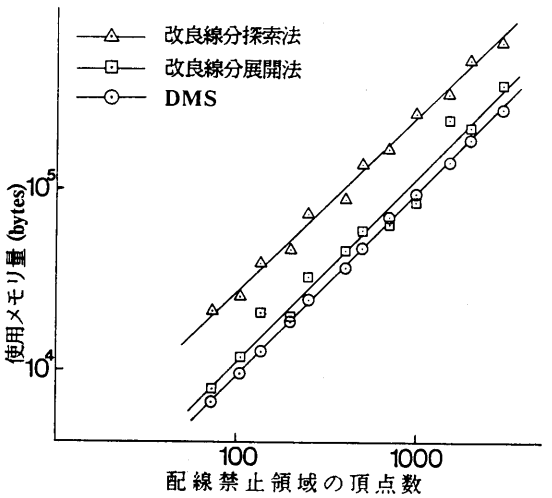


図７：配線禁止領域の頂点数と使用メモリ量の関係

て行う escape line の発生に要する処理時間とはほぼ同等なものとなっている．

使用メモリの量は双方とも配線禁止領域の頂点数に比例しているが，改良線分探索法の使用メモリが改良線分展開法と比較して大きい．これは，改良線分探索法が常に配線領域全体に escape line などを作成しているのに対して，改良線分展開法は径路探索の際に順次 active line を発生し，径路を発見した時点で active line の発生を終了するためである．

以上の結果と考察より，改良線分展開法は改良線分探索法よりも処理時間の点ではやや劣るが同程度の配線処理能力を持つことが示された．したがって，少ないメモリ内で処理を行っていることと高い柔軟性を考慮すれば，改良線分展開法は非常に有効な配線手法であることがわかる．

6．あとがき

本稿では一層配線問題に焦点をあて，O（n）の記憶容量の下で1ネット当たりO（k$\log^2$n）の手間で径路を求める改良線分展開法を提案した．ここで，nは配線領域を図形としてとらえた際の頂点数，kは径路探索時に探索した頂点の数である．また，計算機実験を通じてその有効性を確認した．改良線分展開法は従来の迷路法や線分探索法の置き換えとしてだけではなく，その柔軟性により，バス配線をはじめ，部品の端子間隔が多様化し表面実装等の新しい技術が用いられるプリント基板や，多種の配線幅や複雑な設計規則を扱うアナログＬＳＩのレイアウト設計に有効である．また改良線分展開法は処理時間の点では改良線分探索法と同等ではあるが，将来の専用ハードウェア[10]の利用を前提とすれば，使用メモリが少ない点，処理が単純であるという点においてより有望であると思われる．

最後に今後の研究課題をあげる．

(1) コスト関数

2節のコスト関数を用いる場合にはa，b，cの値を検討する必要がある．また，active line が長くなり，これに対してひとつのコストを与えることがふさわしくない場合には，探索時にある一定の距離以上の探索を行わないようにするか，active line 上で可変なコストを導入することが必要となる．

(2) 多層配線

提案した手法を多層配線モデルを扱えるように拡張する．これにともない，ビアや各層の方向性（縦横原則など）を考慮したコスト関数の考察を行う．

(3) 図形探索用データ構造

本稿ではヒープ探索木を用いたが，他のデータ構造を検討し，必要があれば変更を行う．

謝辞　本研究は財団法人大川情報通信基金：助成番号63-13（昭和63年度）「計算幾何学アルゴリズムのハードウェア化に関する研究」の助成のもとに行われたものである．

参考文献

[1] K.L.Clarkson, S.Kapoor, and P.M.Vaidya: "Rectilinear Shortest Paths through Polygonal Obstacles in O(n (log n)^2) Time", Proc. 3rd Annual Symp. on Computational Geometry, pp.251-257 (1987).

[2] A.Margarino, A.Romano, A.DeGloria, F.Curatelli, and P.Antognetti: "A Tile Expansion Router", IEEE Trans. on CAD, Vol. CAD-6,No.4(1987).

[3] 佐藤政生，坂中二郎，大附辰夫："タイル平面に基づく最小曲がり径路探索アルゴリズム"，情処論，Vol.30, No.2, pp.226-233 (1989).

[4] W.Lipski: "AN O(n log n) Manhattan Path Algorithm",Inf. Process. Lett., Vol.19, No.2, pp.99-102 (1984).

[5] Wu.Ying Fung, P.Widmayer, M.D.F.Schlag, and C.K.Wong: "Rectilinear Shortest Paths and Minimum Spanning Trees in the Presence of Rectilinear Obstacles", IEEE Trans. on Comput. Vol.C-36, No.3, pp321-331 (1987).

[6] 鈴木敬，小島直仁，佐藤政生，大附辰夫："線分探索法の改良とその評価"，信学論，Vol.J72-A, No.2, pp.359-366 (1989).

[7] W.Heyns, W.Sansen, and H.Beke: "A Line-Expansion Algorithm for the General Routing Problem with a Guaranteed Solution", Proc. 17th DA Conf., pp.243-249 (1980).

[8] 乗松誠志，金整範，佐藤政生，大附辰夫："バスの最小ビア配線と置換グラフ"，信学春季全国大会，No.A-268, p.1_271 (1989).

[9] E.M.McCreight: "Priority Search Trees", SIAM J.Comput., Vol.14, No.2, pp.257-276 (1985).

[10] 鈴木敬，大附辰夫："連想メモリを用いたＶＬＳＩ設計用図形処理ハードウェア"，信学論，Vol.J72-A, No.3, pp.550-560 (1989).