組合せ論的および数理計画法的ネットワーク算法に関する研究

吉村 猛

内容梗概

本論文は，筆者が昭和49年日本電気株式会社入社以来，中央研究所およびC&Cシステム研究所で行ってきたネットワーク型組合せ論的および数理計画法的な算法に関する研究成果をまとめたものである。

VLSIをはじめとする科学技術の進歩は目覚しく，システムの大規模化，複雑化が加速的に進行している。このような大規模システムの解析および設計問題は人間の処理できる範囲をはるかに超えており，計算機による支援が不可欠となっている。ところが，現実に存在する問題のほとんどは，計算機が最も苦手とする組合せ問題であり，実用規模の問題に対して現実的な時間内に解を求めるのは極めて困難である。従って，近似的なヒューリスティックな手法で解決せざるをえないのが現状である。

このような問題の多くはネットワーク型の構造，すなわちノードとアーク及びそれらに対する付加情報の形で表現することができる。本研究では，ネットワーク型大規模システムの代表的な例としてVLSI・CADと水道網の設計および解析問題を取上げ，これらの問題に対して，理論と応用の面から検討を加える。そして，ネットワークの基本アルゴリズムを応用した独自の新しい手法を考案することによって，大規模なネットワーク型組合せ問題，数理計画問題の効率的処理を実現することを目標とする。

本論文は全6章から構成される。

第1章ではネットワーク型組合せ論的および数理計画法的な算法に関するこれまでの研究と課題についてのべ，本研究の目的を明らかにするとともに，研究内容と成果について概説する。

第2章ではVLSIのレイアウトCADにおける代表的な問題であるチャンネル配線問題を取り上げ，グラフ理論的な立場から考察する。そして，配線位置に関する制約を表すグラフ（上下制約グラフ）を用い，そのグラフを逐次変形することにより配線経路を決定する方法を提案する。この手法では解を求める過程において解選択の自由度を保ったまま処理することができ，解の大幅な改善が可能となる。また，同じチャンネル配線問題に対して，最長経路アルゴリズムにより各水平トラックごとの最適な割り当てを繰り返し，チャンネル全体の配線を行なうアルゴリズムを提案する。計算機実験によりこれらの手法が短時間にしかもほとんどの場合最適解を発見すること、および従来の手法に比べ計算時間，得られる解の良さの点で優れていることを示す。

第3章ではレイアウトコンパクション問題を考察する。ここではまず，この問題が線形計画問題として定式化されることを示し，さらにグラフ上で基本カットセット，基本ループに関する2つの性質を満たす木の発見問題に帰着されることを証明する。そしてこの木を木の初等変換の繰り返しにより求める方法を提案し，この手法が線形計画問題の代表的解法であるシンプレックス法に対応することを示す。さらに，上記の総配線長最小コンパクション問題を最小コストフローアルゴリズムを用いて解く方法を提案し，その手法の正当性を証明する。

第4章では論理設計CADにおいて最も重要である論理自動合成問題を取り上げ，知識処理とアルゴリズムに基づく解法について考察する。この問題では処理の高速性と製造プロセスの変化

に追随する柔軟性が要求される。そのため，知識処理とアルゴリズムを組み合せる事によって解く方法を提案する。変換規則の適用に対して分枝限定法を用いる事により高速な処理を実現している。既存の計算機の中で使用されている回路に対して人手設計と比較評価した結果，ほとんど同等の解が数分以内に得られたことを示す。

第5章では水道網解析問題をとりあげる。この問題は流量および圧力を電流および電圧，ポンプを電圧源，取水点および需要点を電流源，管路を非線形抵抗にそれぞれ対応させると非線形回路網解析問題となる。この問題に対して，管路の非線形特性を区分線形近似し，その近似精度を動的に更新して解く手法を提案する。

第6章ではこれらネットワーク型の構造を有する諸問題に対して本研究で得られた成果を要約し，今後の研究への期待を述べる。

関連発表論文

I 論文誌発表論文

1. Takeshi Yoshimura and Ernest S. Kuh: "Efficent Algorithms for Channel Routing", IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, Vol. CAD-1, No.1, pp.25－35 (January 1982).
2. 吉村猛: "区分線形近似による管網解析手法", 電子通信学会論文誌, Vol. J66-A, No.4, pp.297–304 (1983-4).
3. 吉村猛: "ネットワーク問題におけるシンプレックス法", 電子情報通信学会論文誌, Vol. J70-A, No.2, pp.156–163 (1987-2).
4. Satoshi Goto, Tatsuo Ohtuski and Takeshi Yoshimura: "Sparse Matrix Techniques for the Shortest Path Problem", IEEE Trans. Circuits and Systems, Vol. CAS-23, No. 12, pp.752–758 (December 1976).
5. 小橋知子, 吉村猛: "水運用計画における予測問題とタンクモデル", 情報処理学会論文誌, Vol. 24, No.4, pp.406–413 (1983-7).
6. Masaki Ishikawa, Tsuneo Matsuda, Takeshi Yoshimura and Satoshi Goto: "Compaction-based Custom LSI Layout Design Method", IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, Vol. CAD-6, No.3, pp.374–382 (May 1987).

II 国際学会発表論文(査読付)

1. Takeshi Yoshimura and Ernest S. Kuh: "Some Algorithms for Channel Routing", Proc. International Symposium on Circuits and Systems (ISCAS), p.955 (May 1980).
2. Takeshi Yoshimura: "An Efficient Channel Router", Proc. 21st Design Automation Conference, pp.38–44 (June 1984).
3. Takeshi Yoshimura: "A Graph Theoretical Compaction Algorithm", Proc. International Symposium on Circuits and Systems(ISCAS), pp.1455–1458 (June 1985).
4. Takeshi Yoshimura and Satoshi Goto: "A Rule-Base and Algorithmic Approach for Logic Synthesis", Proc. International Conference on Computer-Aided Design (ICCAD), pp.162–165 (November 1986).
5. Satoshi Goto, Tatsuo Ohtsuki and Takeshi Yoshimura: "A Shortest Path Calculation Program based on Code Generation Technique", Proc. 13th Allerton Conference, pp.431–438 (October 1975).
6. Masaki Ishikawa, Tsuneo Matsuda, Takeshi Yoshimura and Satoshi Goto: "An Automatic Compaction Method for Building Block LSIs", Proc. of International Symposium on Circuits and Systems (ISCAS), pp. 203–206 (June 1985).

7. Ryuichi Takahashi, Takeshi Yoshimura and Satoshi Goto: "A VLSI Architecture Evaluation System", Proc. International Conference on Computer Design (ICCD), pp.60–63 (October 1986).

8. Masaki Ishikawa and Takeshi Yoshimura: "A New Module Generator with Structural Routers and Graphical Interface", Proc. International Conference on Computer-Aided Design (ICCAD), pp.438–439 (November 1987)

9. Kazutoshi Wakabayashi and Takeshi Yoshimura: "A Resource Sharing Control Sysnthesis Method for Conditional Branches", Proc. International Conference on Computer-Aided Design (ICCAD), pp.62–65 (November 1989).

10. Masato Edahiro and Takeshi Yoshimura: "New Placement and Global Routing Algorithms for Standard Cell Layouts", 27th Design Automation Conference, pp.642–645 (June 1990).

11. Masato Edahiro and Takeshi Yoshimura: "Minimum Path-Length Equi-Distant Routing", Proceedings of Asia-Pacific Conference on Circuits and Systems(APCCAS), pp.41–46 (December 1992).

12. Takashi Fujii, Yoko Miwa, Tsuneo Matsuda and Takeshi Yoshimira: "A Multi-Layer Channel Router with New Style of Over-the-Cell Routing", Proc. 29th Design Automation Conference, pp.585–588 (June 1992).

13. Yuichi Nakamura and Takeshi Yoshimura: "A Partitioning-based Login Optimization Method for Large Scale Circuits with Boolean Matrix", Proc. 32nd Design Automation Conference, pp.653–657 (June 1995).

目次

第 1 章 序論 **1**

第 2 章 チャンネル配線 **3**

2.1 緒言 3
2.2 問題の定式化 3
2.3 基本算法 9
2.4 算法の改良 12
2.5 高速算法 18
2.6 実験結果と考察 23
2.7 結言 25

3 章 コンパクション **26**

3.1 緒言 26
3.2 問題の定式化 27
3.3 グラフ理論的考察 30
3.4 木の初等変換による算法 35
3.5 最小コストフローアルゴリズムを用いた算法 37
3.6 応 用 47
3.7 結言 48

第 4 章 論理合成 **49**

4.1 緒言 49
4.2 問題の定式化 50
4.3 論理合成算法 51
4.4 算法の高速化 55
4.5 考察 57
4.6 結言 58

第 5 章 管路網解析 **59**

5.1 緒言 59
5.2 問題の定式化 60
5.3 区分線形近似による管網解析手法 62
5.4 疎行列演算手法 67
5.5 実験結果と考察 69
5.6 結言 72

第 6 章 結論 73

謝辞 75

参考文献 76

第1章 序論

VLSIを始めとする科学技術の進歩は目覚しく，システムの大規模化，複雑化が加速的に進行している。このような大規模システムの解析および設計問題は人間の処理できる範囲をはるかに超えており，計算機による支援が不可欠となっている。これらの問題の内のいくつかは数理計画算法により多項式時間で解くことができるが，現実に存在する問題のほとんどは，計算機が最も苦手とする組合せ問題である。そのため最適解を求めるのに問題規模の指数関数の時間を要すると考えられ，実用規模の問題に対して現実的な時間内に解を求めるのは極めて困難である。従って，近似的なヒューリスティックな手法で解決せざるをえないのが現状である。従来，これらの問題に対して各々の問題ごとにヒューリスティックな近似解法が提案されてきたが，力まかせに解く方法も多く，大規模システムを扱う問題に対しては，得られる解の精度，処理速度，使用記憶装置の量などの点で必ずしも十分とは言えなかった。

このような問題の多くはネットワーク型の構造，すなわちノードとアーク及びそれらに対する付加情報の形で表現することができる。一方，ネットワーク上で最短経路，最小木，最小コストフロー，最大フローなど計算するための基本アルゴリズムは古くから研究され，効率的な手法が提案されている[1]。本研究では，ネットワーク型大規模システムの代表的な例としてVLSI・CADと水道網の設計および解析問題を取上げ，これらの問題に対して，理論と応用の面から検討を加える。そして，ネットワークの基本アルゴリズムを応用した独自の新しい手法を考案することによって，大規模なネットワーク型組合せ問題，数理計画問題の効率的処理を実現することを目標とする。

本研究ではまず，VLSIレイアウトCADにおける代表的な問題であるチャンネル配線問題[2–5]を取り上げ，グラフ理論とその応用の面から検討を加えた。そして，二つの新しい解法を提案し，解の精度と処理時間の観点から有効性を確認した。また，同じくVLSIレイアウトの代表的な問題であるコンパクション問題[9–21]について考察した。これは線形計画問題として定式化されるが，そのまま解く場合，行列演算を伴うため計算時間の点で不利である。ここではこの問題がグラフ上である性質を持つ木を求める問題に帰着されることを示し，グラフ理論的に解くための二つの方法を提案した。次に，LSI論理自動合成問題[22–26]に対して，知識処理とアルゴリズムの組合せによる解法を提案し，既存コンピュータの回路を用いて，人手での設計と比較評価してその実用性を示した。さらに，水道網の解析問題[30, 31]は非線型抵抗回路網の解析問題となるが，この問題に対して，区分線形近似手法[32]に基づく管路網内の流量圧力解析算法を考案し，実際の水道管網に適用して有効性を確認した。

本論文は，全6章から構成される。

第2章ではVLSIのレイアウトCADにおける代表的な問題であるチャンネル配線問題を取り上げる。従来この問題に対しては直感的な近似手法が用いられてきており，得られる解の精度は必ずしも十分なものではなかった。ここでは，配線位置に関する制約を表すグラフ（上下制約グラフ）を用い，その上でネット(ノード)を併合する処理を繰り返して配線経路を決定する方法を提案する。この手法では各ネットが配線される具体的な位置(トラック)は最終段階まで確定する必

要はないため，解の自由度を保ったまま処理することができ，解の大幅な改善が可能となる。計算機実験によりこの手法が短時間にしかもほとんどの場合最適解を発見すること、および従来の手法に比べ計算時間，得られる解の良さの点で優れていることを示す。

また，同じチャンネル配線問題に対して，最長経路アルゴリズムにより各水平トラックごとの最適な割り当てを繰り返し，チャンネル全体の配線行なうアルゴリズムを提案する。この手法では一つのトラックだけ考慮した場合，最適な割り当てが保証される。この手法は第1の手法と処理自体は全く異なるが，必要配線領域がほぼ同等な解を高速に計算可能で，しかも凹凸のあるチャンネルの処理にも適用できるため実用上有効であることを示す。

第3章ではレイアウトコンパクション問題を考察する。従来，面積を最小化する問題については多くの手法が提案されてきたが,配線長の総和を最小化することを目的とする問題に対しては,最適解を効率良く求める手法は知られていなかった。ここではまず，この問題が線形計画問題として定式化されることを示し，さらにグラフ上で基本カットセット，基本ループに関する2つの性質を満たす木の発見問題に帰着されることを証明する。そしてこの木を木の初等変換の繰り返しにより求める方法を提案し，この手法が線形計画問題の代表的解法であるシンプレックス法に対応することを示す。

さらに，上記の総配線長最小コンパクション問題を最小コストフローアルゴリズムを用いて解く方法を提案し，その手法の正当性を証明する。コンパクションに関するこれらの解法は線形計画問題をシンプレックス法を用いてそのまま解くのに比べ，行列演算を行う必要がないためはるかに効率が良いことを特徴とする。

第4章では論理設計CADにおいて最も重要である論理自動合成問題を取り上げ，知識処理とアルゴリズムに基づく解法について考察する。この問題では処理の高速性と製造プロセスの変化に追随する柔軟性が要求される。そのため，知識処理とアルゴリズムを組み合せることによって解く方法を提案する。変換規則の適用に対して分枝限定法を用いることにより高速な処理を実現している。既存の計算機の中で使用されている回路に対して人手設計と比較評価した結果，ほとんど同等の解が数分以内に得られたことを示す。

第5章では水道網解析問題をとりあげる。この問題は水道管網の取水量や需要量，ポンプの圧力等の条件下における管網内の流量および圧力を計算するものである。この問題は流量および圧力を電流および電圧，ポンプを電圧源，取水点および需要点を電流源，管路を非線形抵抗にそれぞれ対応させると非線形回路網解析問題となる。この問題の解法としてニュートン法が有名であるが初期解の与え方によっては解の収束性に問題が生ずるという欠点があった。そこで，管路の非線形特性を区分線形近似し，その近似精度を動的に更新して解く手法を提案する。あわせて，疎行列に対する新しいLU分解手法を提案すループログラム実験の結果，この手法は全ての例題について収束し，計算時間も従来の手法に比べ1/50倍以下であることを示す。

第6章ではこれらネットワーク型の構造を有する諸問題に対して本研究で得られた成果を要約する。そして，本研究の成果がこれらの諸問題に対する処理時間の短縮，解の改善に重要であり，しかも一般の組合せ問題，数理計画問題への広い応用が期待されることを述べる。

第 2 章 チャンネル配線

本章では代表的なレイアウト問題の一つであるチャンネル配線問題を考察する。まず、各ネットの配線位置に関する制約を表すグラフ上でネットの併合を繰り返して解く手法を提案し、計算機実験によりこの手法が短時間にしかもほとんどの場合最適解を発見すること、および従来の手法に比べ計算時間，得られる解の良さの点で優れていることを示す。 次に，同じチャンネル配線問題に対して，最長経路アルゴリズムにより各水平トラックごとの最適な割り当てを繰り返し，チャンネル全体の配線を行なうアルゴリズムを提案する。この手法は第 1 の手法とは全く異なるが，必要配線領域がほとんど同じ解をさらに高速に計算可能であり，しかも凹凸のあるチャンネルの処理にも適用できるため実用上有効であることを示す。

2.1 緒言

LSI のレイアウトにおける配線設計の究極の目標は各モジュール間の結線をできるだけ少ない面積で実現することである。これまで極めて多くの配線手法が提案されてきたが，その中でもチャンネル配線手法は実用上特に重要である。その理由としては，①単純で効率が高いこと，②チャンネル配線幅が可変である時 100%配線の実現が保証されること，③ゲートアレイ方式，ポリセル方式など規則的な構造をもつチップだけでなくカスタム VLSI チップの設計にも適用できること等があげられる。この問題に対しては Left Edge Algorithm とよばれる算法が Hashimoto&Stevens[2]によって最初に提案され，Deutsch 等[3–5]によって改良が加えられた。しかし，これらの算法は得られる解の良さという点で充分なものではなかった。そのため本文では 3 種類の新しいチャンネル配線手法を提案し，計算機実験により評価を行なった。これらの手法はいずれもネットワーク算法を使用しており，計算時間，得られる解の良さの点で既存の手法を大幅にこえることが確認された。

2.2 問題の定式化

上辺と下辺に端子の列をもつ矩形のチャンネルを考える。各端子には 0 から N までの数字が割り当てられている。これらの端子のうち同じ番号 $i(i \leqq N)$をもつ端子はネット i によって結ばれなければならない。これに対して番号が 0 の端子は結ぶ必要はないことを表わす。

配線には 2 層が使用可能であり，全ての水平トラックは第 1 層に，また全ての垂直トラックは第 2 層に置かれると仮定する。従って水平トラックと垂直トラックは電気的に分離される。ネットはこれら水平トラックと垂直トラックの上の経路で結線され，第 1 層と第 2 層間の結線はスルーホールによって実現される。配線要求は図 2.1 に示すネットリストでも表現される。

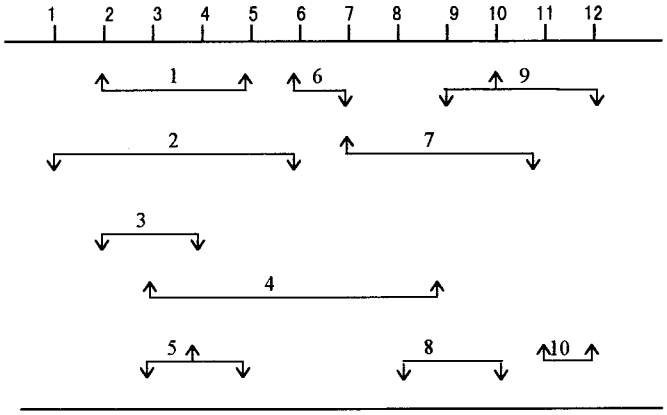


図 2.1 配線要求とネットリスト

図 2.1 において，矢印はネットがチャンネルの上側の端子と下側の端子のいずれにつながるかを表わしている。図 2.2 にこの例に対する一つの配線例を示す。

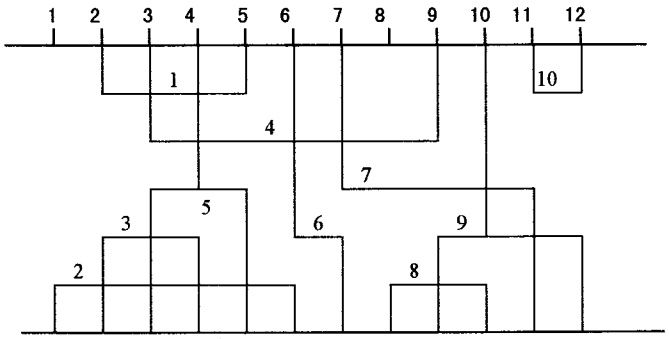


図 2.2 チャンネル配線結果

(1) Cyclic Conflict

図 2.3 の例に注目する。各ネットの垂直成分は同一の垂直トラック上の他のネットの垂直成分と重なってはならないのでネット 1 およびネット 2 の水平成分の位置に関する制約が生じる。例えば一番左の列に注目すると，ネット 1 の水平成分はネット 2 の水平成分のそれよりも上に置か

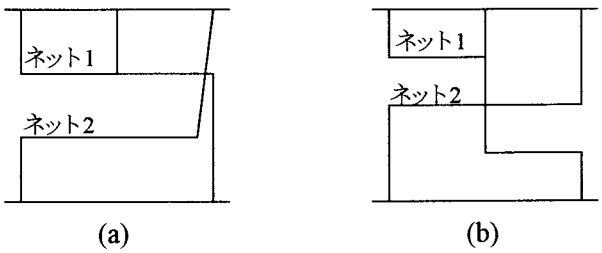


図 2.3 Cyclic Conflict の例

れなければならない。一方，右端の列を考えると，同じ理由でネット 2 はネット 1 よりも上になければならない。すなわち，このような結線要求は図 2.3(b)に示すように，ネットの水平成分を

いくつかに分解しない限り実現不可能である。しかしこのような状況はモジュールの端子の位置を変更することで多くの場合回避することができる。そこで，本文では配線要求は常に実現可能である，すなわち，ネットリストにはCyclic Conflictは存在しないと仮定する。

(2) ドッグレッグ

問題の目的関数は結線要求を実現するのに使用する水平トラック数を最小にすることである。図2.4(a)の例を考える。ネットの水平成分の分割が許さない場合にはこれが最適解である。しかし，水平成分を分割すれば同じ結線要求は同図(b)のように2つの水平トラックで実現できる。このようなネットの水平成分の分割を“ドッグレッグ”とよぶ。これは，前述のようにcyclic conflictを解消するために用いられるほか，水平トラック数を減少させるためにも用いられる。なお，本文ではドッグレッグを行なう場合，ネットの水平線分の分割はそのネット上の端子位置だけに限定する。

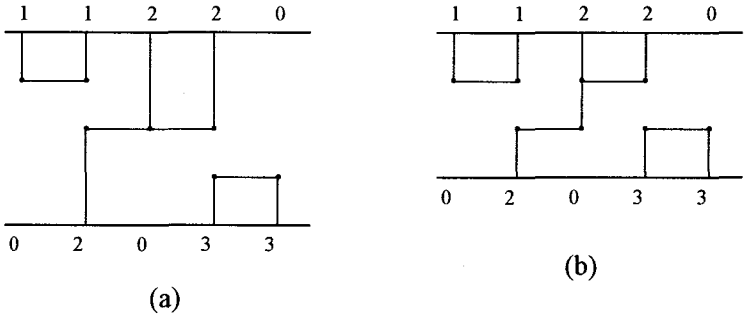


図2.4 ドッグレッグによる配線の利点

(3) 垂直制約グラフ

すでに述べたように，二つのネットの垂直成分は同一垂直トラック上で重なってはならない。そこで，もし，ネットの水平成分の分割(ドッグレッグ)を許さないとしたならば，チャンネルの上辺にある端子につながるネットの水平成分はその列の下辺にある端子につながるネットの水平成分よりも上に置かれなければならない。

すべてのネットに対してこのような関係は次に垂直制約グラフG_vによって表わすことができる。グラフG_vにおいて，各ノードは一つのネットに対応し，ノードaからノードbに向う枝はネットaはネットbよりも上に置かれなければならないことを表わす。従って，もしこのグラフ中にサイクルがあれば，配線要求は少なくとも1本の水平成分を分割しない限り満たすことができない。例えばサイクルa→b→c→aはネットaがそれ自身より上に置かれなければならないことを意味する。しかし，逆に，サイクルがなければその配線要求は常に実現可能である。図2.1のネットリストに対する垂直制約グラフを図2.5に示す。

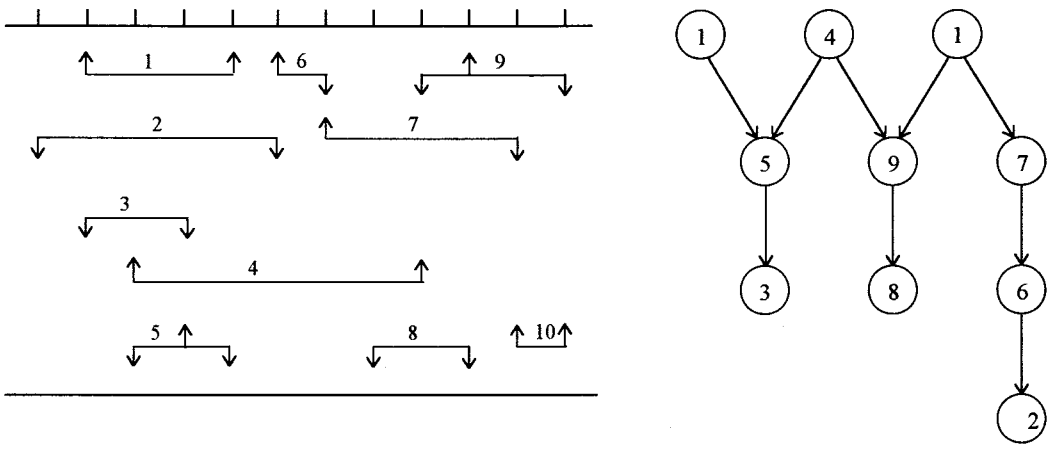


図 2.5 垂直制約グラフ

(4) 先祖(Ancestor)と子孫(Descendent)

垂直制約グラフ上でノード i からノード j に向う有向経路がある時，ノード i をノード j の先祖(j は i の子孫)とよぶ。

(5) 水平成分のゾーン表現

ネットの水平成分はその左端と右端の端子の位置によって決まる。$S(i)$ を列 i を横切るネットの集合とすると，相異なるネットの水平成分は同一の水平トラック上で重なることはできないので，$S(i)$ の任意の 2 つのネットは同一水平トラックに置くことはできない。この条件は全てのトラックで満たされなければならない。しかし，水平成分どうしの重なりを扱かう場合，全ての列を考える必要はない。すなわち，ある列 i に対する $S(i)$ が他の列 j の $S(j)$ に対する部分集合になっている場合，明らかに列 i は考慮しなくてもよい。そこで $S(i)$ が極大集合となる列 i に対して 1 から順に自然数を割り当て，これらの列をゾーン 1，ゾーン 2，・・・・と定義する。$S(i)$ の要素数を局所配線密度(Local Density)とよび，そのうち最大のものを最大配線密度(Maximum Density)と呼ぶ。

表 2.1 各列を横切るネット集合とゾーン

列	S(i)	ゾーン	列	S(i)	ゾーン
1	2		**7**	4 6 7	3
2	1 2 3		**8**	4 7 8	
3	1 2 3 4 5	1	**9**	4 7 8 9	4
4	1 2 3 4 5		**10**	7 8 9	
5	1 2 4 5		**11**	7 9 10	5
6	2 4 6	2	**12**	9 10	

表 2.1 は図 2.1 のネットリストに対する $S(i)$ を表わしている。この表において，$S(1)$および $S(2)$ は $S(3)$の部分集合となっているので，水平線分の重なりを考慮する場合，$S(1)$および $S(2)$を考える必要はない。そこで，列 3，6，7，9，10 をそれぞれゾーン 1～5 とする。このゾーン表現を図示すると図 2.6(a)となる。

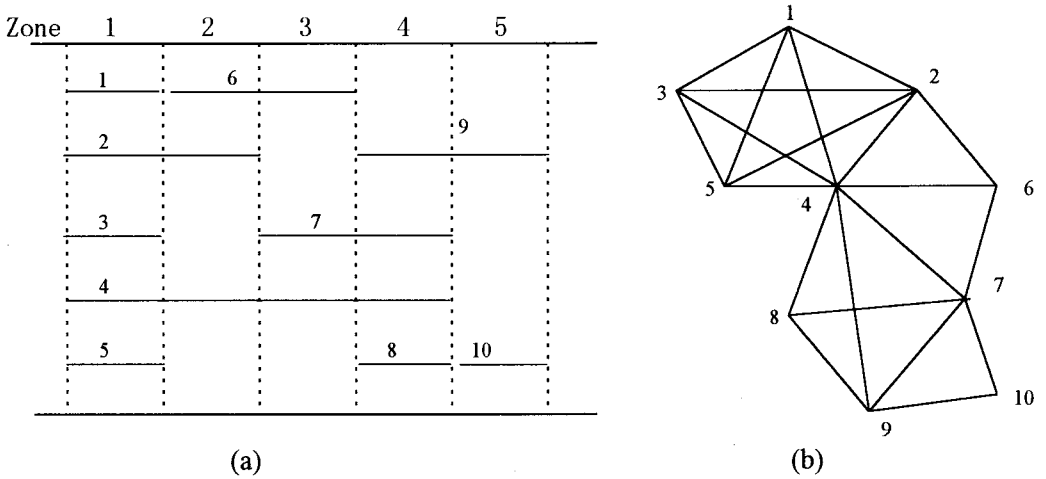


図 2.6 ゾーン表現とインターバルグラフ

ゾーンはネットの水平成分に基づくインターバルグラフ[6, 7]を使えばより正確に定義できる。グラフ $G(V, E)$をネットの集合に対するインターバルグラフとする。ここで $v_i \in V$ はネット n_i に対応し，ネット n_i と n_j の水平成分に重なりがある時，かつその時に限り $(v_i, v_j) \in E$ となるものとする。図 2.1 のネットリストに対応するインターバルグラフを図 2.6(b)に示す。インターバルグラフ上ではゾーンは極大クリークにより定義でき，そのクリーク数(最大クリークのノード数)が最大配線密度となる。

このように，チャンネル配線問題は垂直制約グラフとゾーン表現によって完全に記述することができる。

(6)ドッグレッグへの対処

これまでは垂直制約グラフとゾーン表現はドッグレッグなしの場合に対して定義されていた。しかし，ネットの代りにサブネットを用いることによって前記のドッグレッグ問題に対しても適用可能となる。ここで，サブネットとは，ネットを端子の位置で分解したもので，その水平成分はネットの隣り合う端子の間の部分に対応する。図 2.7 はサブネットおよび極大クリークを与えるカットの例を示したものである。図 2.8 はドッグレッグなしの場合のゾーン表現および垂直制約グラフである。一方ドッグレッグを考慮する場合にはネットの代りにサブネットを考えればよく，この例に対する垂直制約グラフは図 2.9 のようになる。

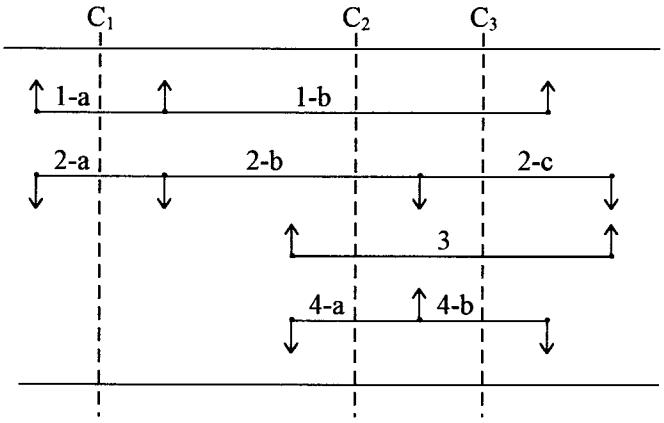


図 2.7 サブネットとカットの例

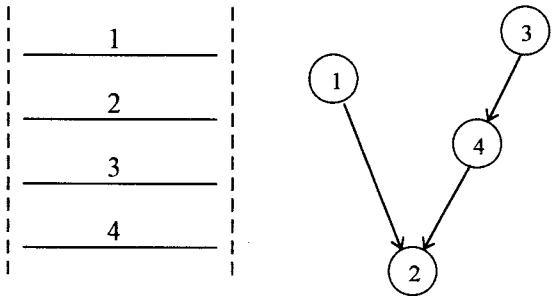


図 2.8 ドッグレッグなしの場合

なお，このようにサブネットを考えることによりドッグレッグを処理できるが，実際の問題を解くにあたってはサブネットの数が非常に多くなり，計算時間，使用記憶装置量の点で不利となる。そのため，前処理として，サブネットをまとめる処理を考える。この処理を完全に行なうとドッグレッグなしの場合に帰着されてしまうので，“2 つのサブネットをまとめても垂直制約グラフ上で 2 つのネットいずれかを通る最長経路の長さが増加しない”という条件のもとでサブネットどうしをまとめて実質的なネットの数をへらしている。

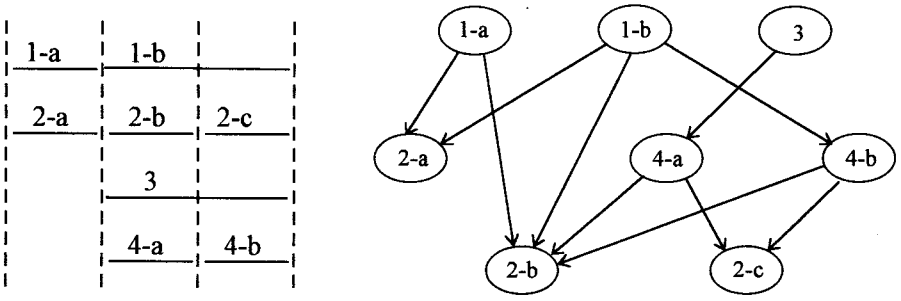


図 2.9 ドッグレッグありの場合

2.3 基本算法

チャンネル配線問題の最適解を求めるのは非常に困難である。そのため，本文では，妥当な計算時間内で近似最適解を求めるヒューリスティックアルゴリズムを提案する。

(1)ネットの併合

アルゴリズムについて述べる前に次の定義を行なう

定義: ネットの併合

ネットi，ネットjは次の条件を満たすものとする

① ゾーン表現において両者の水平線分は重ならない。

② 垂直制約グラフにおいて，ノードiとノードjの間に有向パスは存在しない。
(すなわち，両者の間に上下制約はない)

このとき，“ネットiとネットjの併合”とは

① 垂直制約グラフのノードiとノードjを新しい一つのノードi・jに縮約する。

② ゾーン表現において，ネットiとネットjをネットi・jでおきかえ，
ネットi，ネットjを含む連続した領域をネットi・jの領域とする。

例: 図2.1の例において，ネット6とネット9は併合の対象となる。ネット6とネット9の併合の結果，垂直制約グラフとゾーン表現は2.10(b)および図2.10(c)となる。この更新された垂直制約グラフとゾーン表現はネット6とネット9が併合された図2.10(a)のネットリストに対応している。この併合操作はネット6とネット9が同じトラックにおかれることを意味している。しかし，それがどのトラックであるかは未定である。

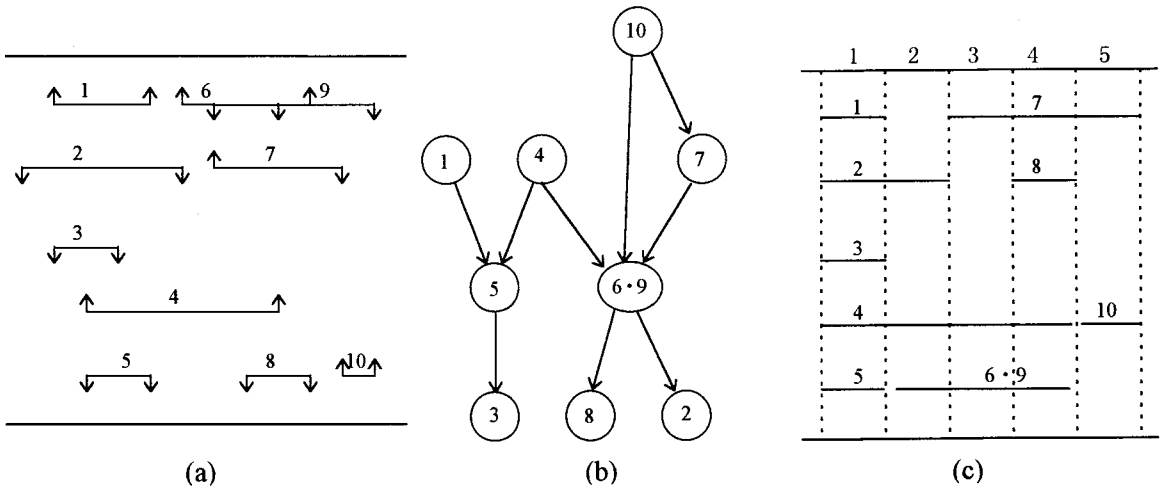


図2.10 更新後の表現

この併合に関して次の補題が成立する。

補題 1: ネット併合前の垂直制約グラフがサイクルを含まない時，併合後の垂直制約グラフもサイクルを含まない

(2)アルゴリズム

最初の垂直制約グラフがサイクルを含まないため，サイクルを生成することなくネットの併合操作を繰り返すことが出来る。次のアルゴリズムは，ゾーン表現を利用してネットの併合を繰り返すものである。

アルゴリズム#1

Proc. Algorithm#1(Z_s, Z_t)

begin

a1: L={ }

a2: for $z = Z_s$ to Z_t do

a3: $L = L$ + {ゾーン z を右端とするネット};

a4: R = {ゾーン z+1 を左端とするネット};

a5: L と R のネット集合を併合する。ただし，そのさい垂直制約グラフ上の最長経路の増加をできるだけおさえる。

a6: $L = L - \{n_1, n_2, \cdots\}$，ただし，$n_j$ (j=1,2,・・・)はステップ a5 で併合されたネットとする。

end

end

垂直制約グラフ上のパス n_1–n_2–n_3–・・・–n_k が存在する場合，n_1, n_2, n_3, ・・・, n_kのどの二つのネットも同じトラックにおくことは出来ない。従って，垂直制約グラフ上の(ノード数の意味での)最長経路の長さが k であるとすると，配線を実現するのに少なくとも k トラック必要となる。そのため，ネットを併合する時は最長経路の増加を出来るだけおさえる必要がある。

図 2.11 に上記のアルゴリズムによる上下制約グラフの変化を示す。まずネット 5 とネット 6 が併合され，ついでネット 1 とネット 7 が，さらにネット 5・6 とネット 9 とつづき，最後にネット 4 とネット 10 が併合されている。その結果図 2.11(e)のグラフが得られる。この段階で，グラフの各ノードを水平トラックに割り当てる。例えば，ネット 10・4 をトラック 1 に，ネット 1・7 をトラック 2 に，ネット 5・6・9 をトラック 4 に，ネット 2(またはネット 3・8)をトラック 4 に，そしてネット 3・8(またはネット 2)をトラック 5 に割り当てる。図 2.11(f)にこの割り当てに対応する配線結果を示す。

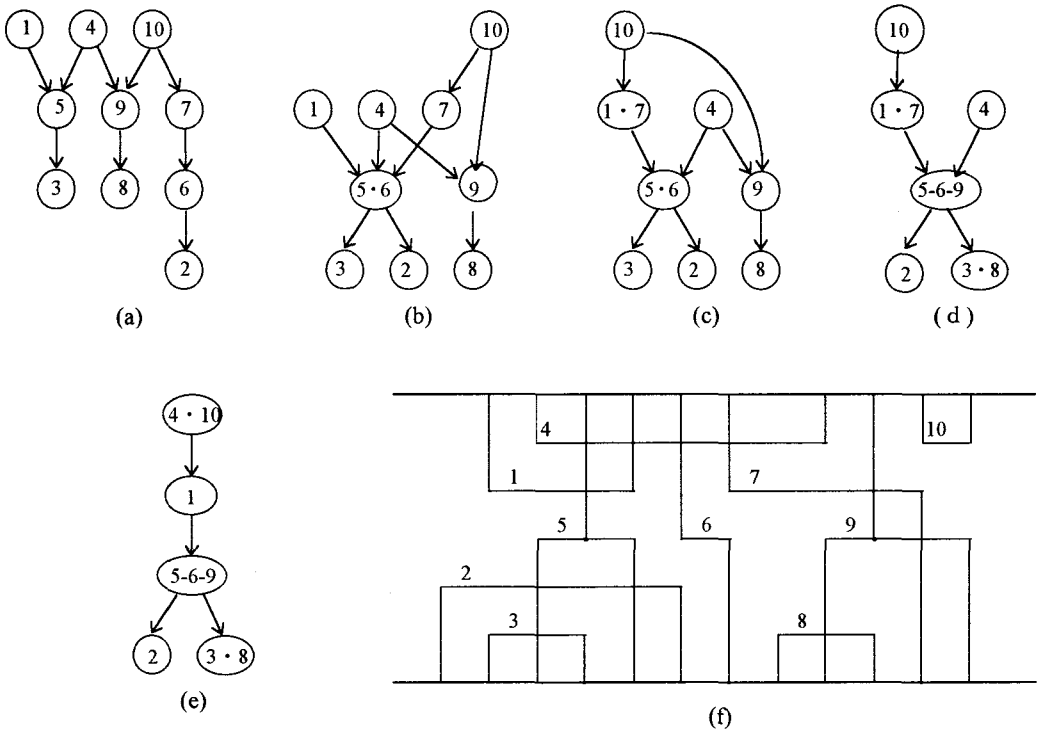


図 2.11　アルゴリズム#1 の説明

(2) 最長経路長の最小化

上記アルゴリズムでもっとも重要なのはステップ a5 におけるネットの併合処理の部分である。次に，このステップに対する一つの実現法を示す。

まず，P，Q，$u(n)$および$d(n)$をそれぞれ

① $P=\{n_1, n_2, \cdots\cdots, n_p\}$，$Q=\{m_1, m_2, \cdots\cdots, m_q\}$ $(p \geqq q)$: 併合すべきネットの集合

② $u(n)$; $n \in P \cup Q$: 垂直制約グラフ上でノード n を終点とする最長経路長 +1

③ $d(n)$; $n \in P \cup Q$: 垂直制約グラフ上でノード n を始点とする最長経路長 +1

とする。

[例]

Q={6,7}, P={1,3,5}

$u(1)=1$, $u(3)=3$, $u(5)=2$, ・・・・・

$d(1)=4$, $d(3)=2$, $d(5)=2$, ・・・・・

目的は併合後の最長経路長を最小化することである。しかし厳密な解を求めるのに要する時間は膨大となるので，ここでは近似的なアルゴリズムを提案する。 まず，集合 Q から最長経路上のノード m を選ぶ(最長経路上のノードが複数個ある時は経路の始点または終点からもっとも遠いものを選ぶ)。つぎに，集合 P の中から，併合による最長経路長の増加が最小となるノード n を

選ぶ。そのような候補が複数個ある時は $u(n)+d(n)$ が小さく，かつ $u(m)/d(m)$ の値が $u(n)/d(n)$ の値に出来るだけ近いノードを選ぶ。

具体的には，次の手順で m と n を決定する。

①次式を最大とする $m(\in Q)$ を選ぶ。

$$f(m) = C_\infty * \{u(m) + d(m)\} + \max\{u(m), d(m)\}, \quad C_\infty >> 1$$

② m が決定された時，次式を最小とする $n \in P$ を選ぶ。

$$g(n,m) = C_\infty * h(n,m) - \{\sqrt{u(m)*u(n)} + \sqrt{d(m)*d(n)}\}$$

ただし $h(n,m)$は「m または n を通る最長経路長が併合によって増加する長さ」，すなわち

$$h(n,m) = \max(u(n), u(m) + \max\{d(n), d(m)\} - \max\{u(n) + d(n), u(m), d(m)\}$$

である。

例えば，前記の例で C_∞=100 とすると，表 2.2(a)を得る。ここで，$f(7)>f(6)$であるから，まずノード 7 を Q からえらぶ。次に，$g(n,7)$を評価すると，$g(4,7)$が最小となるので(表 2.2(b))，アルゴリズムはノード 4 とノード 7 を併合する。

表 2.2　併合ノード選択のための評価関数

(a) ノード m の決定

	P			Q	
	1	3	4	6	**7**
$u()$	1	3	4	2	**3**
$d()$	4	2	1	2	**1**
$f(m)$				402	**403**

(b) ノード n の決定

n	1	3	**4**
$h(n,7)$	2	0	**0**
$g(n,7)$	196.27	−4.41	**−4.46**

2.4 算法の改良

(1)アルゴリズムの概要

アルゴリズム#1 では，ネットはそれらが処理される時，一度併合されたネットの組合せは以後固定である。そのため，後続の併合の障害となることがある。図 2.12 にその例を示す。ゾーン 1 において，アルゴリズム#1 がネット a とネット b およびネット b とネット e をそれぞれ併合したとする(前述のアルゴリズムに従うとこの併合は行なわれないが，説明の都合上このように仮定する)。このとき，垂直制約グラフとゾーン表現はそれぞれ図 2.12(c)および図 2.12(d)の様に更新される。この垂直制約グラフはネット f はネット c，ネット g のいずれとも併合できないことを意味している。これに対して，最初にネット a とネット d およびネット c とネット e を併合すればネット f はネット b と併合可能である。

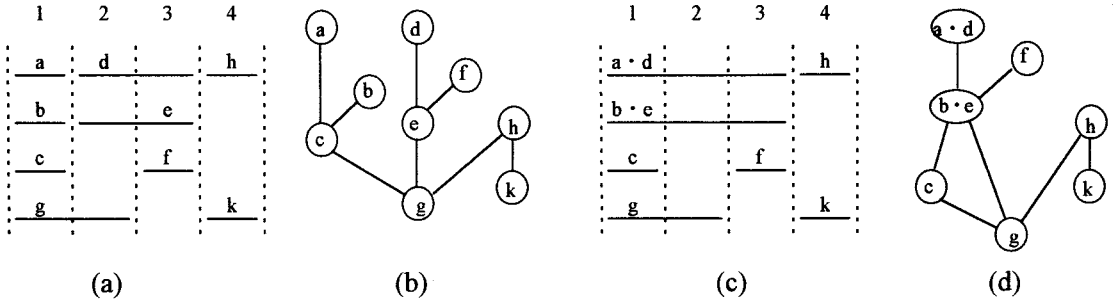


図 2.12 アルゴリズム#1 の問題点

この問題点に対処するため，本文ではアルゴリズム#2 を提案する。このアルゴリズムでは，2 部グラフ $G_h(N, E)$を導入する。ここで N は併合の候補となるノードの集合であり，枝 $(a,b) \in E$ はネット a とネット b が併合可能であることを意味する。ネットの併合はこのグラフ G_h 上のマッチングで表現し，それを順次更新する。この手法を前記の例を用いて説明する。

ゾーン 1(の右端の境界)ではネット d は(同じくネット e も)ネット a，b，c のいずれとも併合可能である。そこで，このアルゴリズムでは，図 2.13(a)に示すグラフ G_h を作成し，一時的な併合をそのグラフ上のマッチングで表現する。しかし，ゾーン表現，垂直制約グラフのいずれもこの段階では更新しない。つづいて，ゾーン 2 の処理に移る。このゾーンではネット g が終端し，また次のゾーンにはネット f の左端が存在する。そこで，ノード g をグラフ G_h の左側に，ノード f を右側にそれぞれ追加する。図 2.12(b)の垂直制約グラフからネット f はネット a，b，c のいずれとも併合可能であるので，3 本の枝を追加し，仮の併合を表すマッチングを更新して，図 2.13(b)を得る。もちろん，このマッチングに対応する併合が垂直制約を満たすことの保証はない(水平制約は自動的に満たされる)。 従って，アルゴリズムはこのグラフ G_h およびその上のマッチングの実現可能性を調べ，図 2.13(c)のように更新する。この処理については後述する。

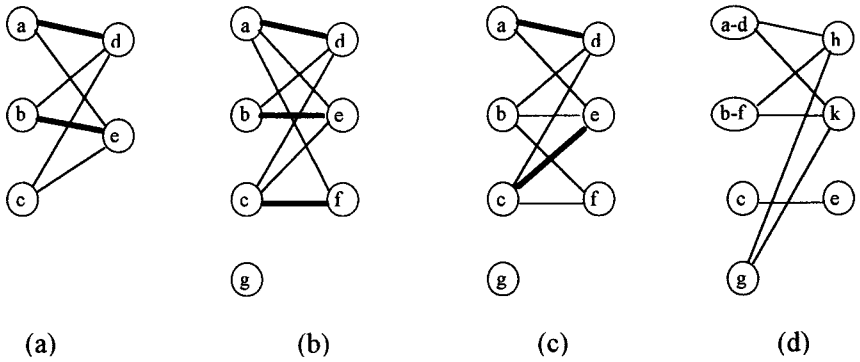


図 2.13 グラフ G_h の更新

ゾーン3でネットdとネットfが終端する。そのため，このゾーンの処理で，ノードdとノードfはグラフG_hの左側に移動させる。その際，現在のマッチングで表されている相手とネットを実際に併合し，グラフG_hを図2.13(d)の様に更新する。また垂直制約グラフもあわせて更新する。なお，ネットeはまだ終端していないため，(ネットcと)併合しない。これらの操作をすべてのゾーンに対して行ない，ネットを併合する。

(2) アルゴリズム#2

Procedure Algorithm#2(Z_s, Z_t)

begin

step1: ゾーンz_sで終端するネットに対応するノードをグラフG_hの左側に追加する。

for $z_i = z_s$ to z_t do

step2: ゾーンz_{i+1}に左端を持つ各ネットn_rに対して，ノードn_rをグラフG_hの右側に追加し，垂直制約を満たす限りグラフG_hの左側のノードとノードn_rの間に枝を設け，グラフG_h上で最大マッチングを求める。

step3: 現在のマッチングに基づくネットの併合の結果，垂直制約が満たされるか否かを調べる。満たされないとき，グラフG_hとマッチングを更新する。

step4: ゾーンz_{i+1}で終端する各ネットn_1について，グラフG_h上のマッチングで表わされる相手のネットn_xと併合する。さらに，グラフG_hからノードn_1およびノードn_xを除去し新たなネット$n_x.n_1$に対応するノードを左側に追加する。

end

end

以下，algorithm#2の各ステップの処理を説明する。

ステップ1

初期化のステップである。アルゴリズムは，処理開始ゾーンz_sで終端するネットに対応するノードをグラフG_hの左側に追加する。この段階では枝はまだ存在しない。

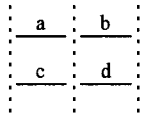
ステップ2

ゾーンz_{i+1}を左端とするネットに対応するノードをグラフG_hの右側に追加し，左側の各ノードに対応するネットのうち併合可能なものとの間に枝を設け，このグラフ上で最大マッチングを求める。

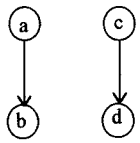
実際には，計算時間と使用記憶装置の量を削減するためノード1個当たりの枝数を制限している(後述のプログラムでは3としている)。枝の選択基準はアルゴリズム#1に関して前節でのべた手法に従っている。

ステップ 3

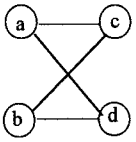
グラフ G_h は水平方向の制約をもとにして作成しているため，グラフ G_h 上の任意のマッチングに対応する解は水平線分の重なりを持たないことは明らかである。しかし，垂直方向の制約は完全には考慮されていないため，垂直制約を満たさない可能性がある。図 2.14 はその例である。



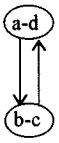
(a) G_h



(c) G_v



(c) G_h



(d) G_v

図 2.14 垂直制約を満たさないマッチング

図2.14(c)のマッチングに対応する併合は垂直制約グラフ上で図2.14(d)のサイクルを生成するため実現可能ではない。したがって，ここでは①与えられたマッチングに対応する併合が垂直制約を満たすか否かを判定し，②満たさない時はマッチングを変更することが必要となる。これら二つの処理を実現するため，その上の任意のマッチングが垂直制約を満たすようなグラフ G_h を導くアルゴリズムを次に提案する。

Algorithm A

グラフ $G_h=(N, E_h)$ とグラフ $G_v=(N, E_v)$ が与えられる。

A1: $E_x=\phi$;

While($N\neq\phi$)do

A2: N_a をグラフ G_v において先祖を持たないノードの集合とする。

A3: グラフ G_h の枝 E_h から次の枝集合 E_a を取り除く。ただし，

$$E_a = \{(i,j) | i,j \in N_a, (i,j) \in E_h\};$$

A4: グラフ G_h において次数(ノードに接続する枝の個数)が 0 となるノードがあるとき，そのノードを v として A6 へ行く。

A5: N_s のうち次数が最小となるノードの一つを v とする。そして，E_y をグラフ G_h においてノード v につながる枝の集合とするとき，

$$E_x = E_x \cup E_y$$

とする。

A6: グラフ G_v およびグラフ G_h よりノード v とそれにつながる枝を取り除く。

end

定理：グラフ G_h における任意のマッチングに対応する併合が実現可能であるための必要十分条件は，アルゴリズムAの終了時における E_x が空であることである。

証明：(十分性) E_x が空の時，グラフ G_h 上の任意のマッチングに対応する併合が実現可能であることを示す。アルゴリズムの i 回目の繰り返し時点におけるノード集合 N_a を $N_a^{(i)}$，ステップA3の枝の除去によって次数が0となる(グラフ G_h の)孤立ノードの集合を $N_s^{(i)}(\subseteq N_a^{(i)})$ とする。

まず，1回目の繰り返しでは，グラフ G_h において $N_s^{(1)}$ のノードと $N_s^{(1)}$ 以外のノードを繰ぶ枝は存在しない。また，$N_s^{(1)}$ ノード相互間には上下関係の制約はないので，グラフ G_h の任意のマッチングに基づく併合に対して(グラフ G_v において)$N_s^{(1)}$ のノードを含むサイクルは生じない。従って，以後これらのノードは考慮の対象から除外することができる。

次に，2回目の繰り返し時点でも同様の理由によりグラフ G_h の任意のマッチングに基づく併合に対して，$N_s^{(2)}$ のノードを含むサイクルは生じないため，以後これらのノードも考慮する必要はないことがわかる。しかも $E_x=\phi$ であるから $N_s^{(2)}\neq\phi$ である。従って，以下同様にこの処理をすべての枝が除去されるまで続けることが出来る。

(必要性) 次に，E_x が空でない時実現可能でないマッチングが存在することを示す。まず，上記と同様の手順で証明を始めるが，最初の枝集合の除去の際孤立したノード集合がないと仮定し，$n_0^{(1)}$ を N_0 の任意のノードとする。この時，グラフ G_h において $n_0^{(1)}$ と N_0 に含まれていないノード $n_d^{(1)}$ を結ぶ枝が少なくとも1つ存在する。もし，$n_d^{(1)}$ が $n_0^{(1)}$ の子孫であればこれらのノードの集合は明らかに実現可能でない。したがって，$n_d^{(1)}$ は $N_0-\{n_0^{(1)}\}$ のノード $n_0^{(2)}$ の子孫であるとする。一方，$n_0^{(2)}$ は枝集合の最初の除去時に孤立していない。そこで同様の理由により，$n_0^{(2)}$ は $N_0-\{n_0^{(2)}\}$ のノード $n_0^{(3)}$ の子孫である $n_d^{(2)}$ と接続されている。ノード数は有限であるから，$n_0^{(1)}$，$n_d^{(1)}$，$n_0^{(2)}$，$n_d^{(2)}$，$n_0^{(3)}$，$n_d^{(3)}$，・・・・となるサイクルが存在する。一般性を失うことなしにサイクルを $n_0^{(1)}$，$n_d^{(1)}$，$n_0^{(2)}$，$n_d^{(2)}$，・・・・，$n_0^{(k)}$，$n_d^{(k)}$，$n_0^{(1)}$ と仮定する。枝集合 $E_s=\{(n_0^{(i)},\ n_d^{(i)})\mid i=1,\ \cdots,\ k\}$ に対応する併合を考え，$n_{0d}^{(i)}$ を $n_0^{(i)}$ と $n_d^{(i)}$ の併合によって出来るノードとする。$n_d^{(i)}$ は $n_0^{(i+1)}$ の子孫であるから，$n_{0d}^{(i)}$ は $n_{0d}^{(i+1)}$ の子孫である $(i=1,\ 2,\ 3,\ \cdots,\ k-1)$。同様に，$n_{0d}^{(k)}$ は $n_{0d}^{(1)}$ の子孫となる。したがって，ノードの併合はグラフ G_v においてサイクルを生成する。(証明終)

補題： グラフ $G_r=(N,\ E_h-E_x)$ 上の任意のマッチングに対応する併合は実現可能である。

この補題に対する例を示す。図2.15(a)はグラフ G_h と G_v を表している。ここでは先祖を持たないノード集合とそれらを接続するグラフ G_h の枝を太い線で示している。これらの枝の除去によりノード d が孤立ノードとなり，それにつながる枝$(d,\ g)$，$(a,\ g)$とともに除去される。図2.15(b)において新しいノードと枝が処理されている。この処理では枝$(a,\ i)$が除去されるが，その結果孤立ノードは発生しない。そこで，図2.15(c)に示すとおり，ノード a および枝$(a,\ h)\ \in E_x$ を除去する。以後，処理は問題なく続行できる。

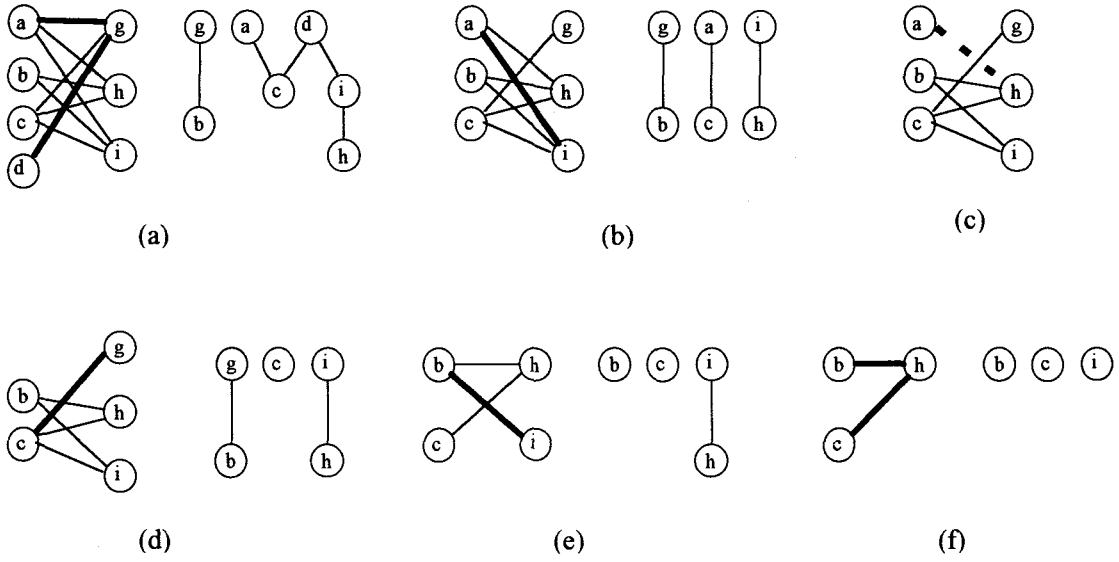


図 2.15 補題の説明

図 2.16(a)に枝集合 E_x を除去したグラフ G_r を示す。太線は任意のマッチングを示している。このマッチング後の垂直制約グラフを図 2.16(b)に示す。図 2.16(a)はある意味ではマッチングで問題を起こす悪い枝$(a,\ h)$が除去されていると考えることも出来る。 アルゴリズムのステップ 3 において，M をマッチング枝とする時，グラフ $G(N,\ M)$にアルゴリズム A を適用することによって垂直方向の制約の検証を行なうことが出来る。明らかに，E_x が空集合の時マッチングは実現可能である。もし，マッチングが実現可能でない時，アルゴリズム A をグラフ $G(N,\ E_h)$に対して適用し，グラフ $G_r=(N,\ E_h-E_x)$上で最大マッチングを再計算する。補題より，この新しいマッチングは実現可能である。

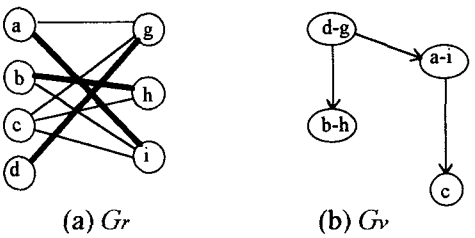


図 2.16 グラフ Gr 上の任意のマッチングと
それに対応する併合後の垂直制約グラフ

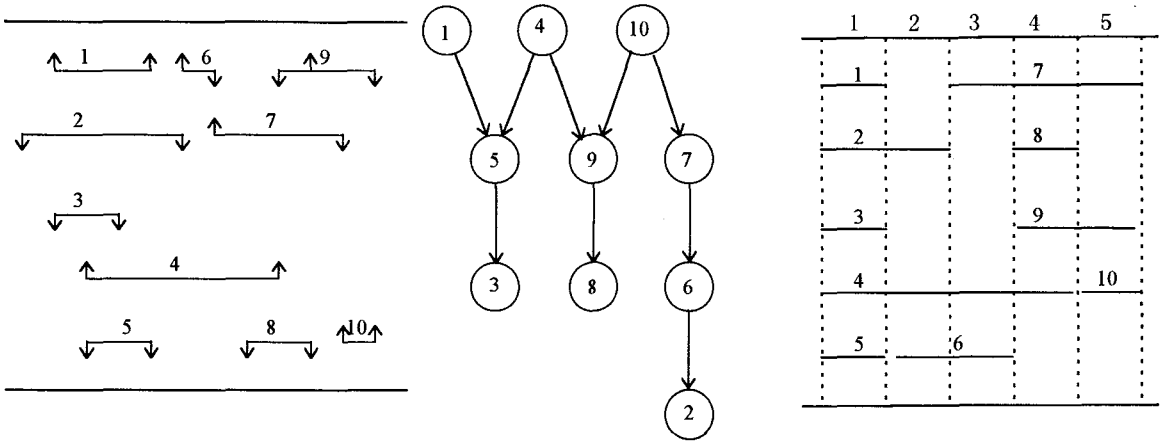
2.5 高速算法

ここでは，同じチャンネル配線問題に対して，最長経路アルゴリズムにより各水平トラックごとの最適な割り当てを繰り返し，チャンネル全体の配線を行なうアルゴリズムを提案する。この手法は前記の手法と全く異なるが，必要配線領域がほぼ同じ解をさらに高速に計算可能である。

(1)Left Edge Algorithm

提案するアルゴリズムについて述べる前にまず，基本的な Left Edge Algorithm(以下 LEA とよぶ)を説明する。このアルゴリズムでは最初に，配線チャンネルの最上位の水平トラックに注目する。このトラックに置くことができるネットは垂直制約グラフ上で先祖をもたないものに限られる。そこでこれらのネットのうちゾーン表現に置いて左端が最も左(のゾーン)にあるものを選び，そのトラックに配置する。次に，そのトラックの残りの部分に配置できるネットがあれば，それらのうち，上記の意味で最も左にあるネットを配置する。この処理を繰り返し，そのトラックに置けるネットがなくなれば垂直制約グラフおよびゾーン表現からそれまでに配置されたネットを取り除き，一つの水平トラックに対する処理を終える。次に，前記トラックの下に新たな水平トラックを用意して同様の処理を行なう。この処理を繰り返して，全てのネットを配置する。

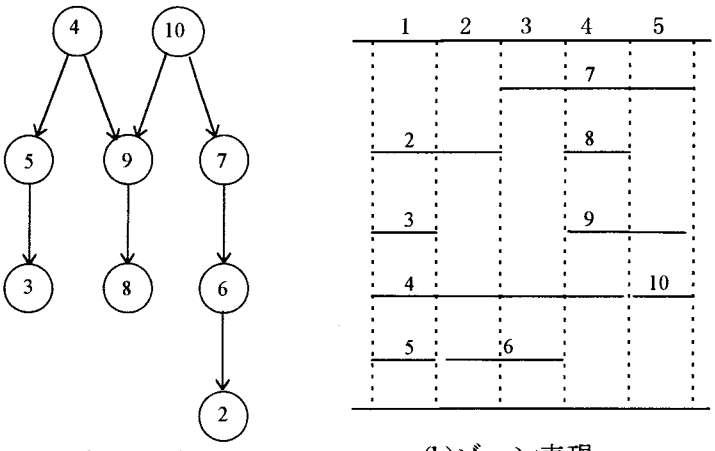
これを図 2.17 の例で具体的に説明する。この例において，最上位の水平トラックにおけるネットは図 2.17(b)の垂直制約グラフからネット 1，ネット 4，ネット 10 であることがわかる。これらのネットのうち図 2.17(c) のゾーン表現上で(左端が)最も左にあるネットはネット 1 またはネット 4 である。ここでは仮にネット 1 を選ぶことにすれば，最上位のトラックにおける次のネットは 10 となる。この結果，最上位のトラックにはネット 1 とネット 10 がおかれる。



(a)ネットリスト　(b)垂直制約グラフ　(c)ゾーン表現

図 2.17 LEA の説明(1)

垂直制約グラフ，ゾーン表現は図 2.18 となる。2 番目のトラックにおけるネットの候補は図 2.18(a)よりネット 4，ネット 7 であるが，図 2.18(b)のゾーン表現よりネット 4 が選択される。この処理を繰り返すことにより最終的に図 2.19 の解が得られる。



(a)垂直制約グラフ　　(b)ゾーン表現

図 2.18 LEA の説明(2)

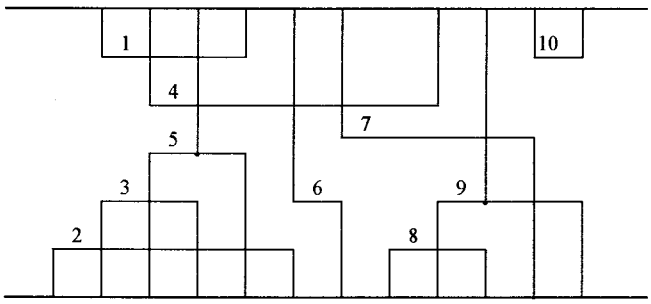


図 2.19　LEA の解

この例では，たまたま最適解が求まったが，図 2.20 の例では最適解が求まらないことがある。すなわち，最初のトラックにおけるネットの候補はネット 1，ネット 2 であるがネット 1 を選んだ場合，図 2.21(a)の解が得られる。この解は 3 本の水平トラックを必要とするが，図 2.21(b)の様に解を構成すれば 2 本の水平トラックで十分である。

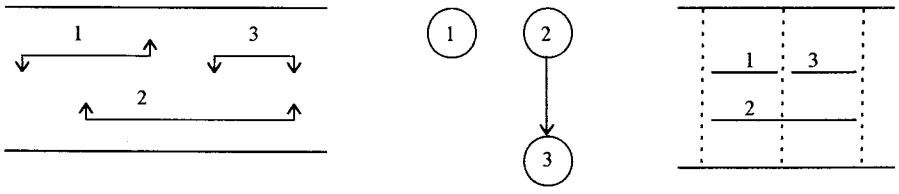
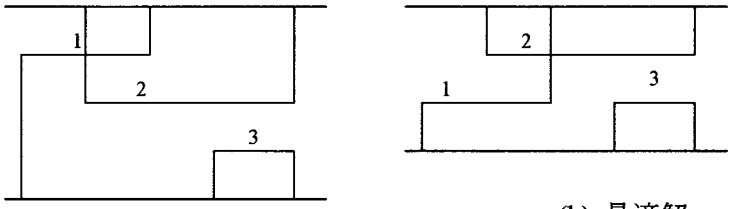


図 2.20　LEA で最適解が得られない例

このように LEA では必ずしも良い解が求まるとは限らない。そこで従来のアルゴリズムでは左端のネットに注目する代りに右端のネットに注目したり，最上位のトラックから処理する代りに最下位のトラックから処理したり，上下交互に処理するなどを行なっていたが，根本的な解決策

とはなっていなかった。次に述べるアルゴリズムは水平トラックの処理順はLEAと同じであるが，1本の水平トラックに対してはある意味で最適になるようネットを割り当てることにより解の改善をはかるものである。



(a) LEAの解　　(b) 最適解

図2.21　　図2.20の最適解とLEAの解

(2)LEAの改良

このアルゴリズムでも基本的には最上位の水平トラックに，垂直制約グラフ上で先祖をもたないネットを割当てる処理を繰り返している。この点はLEAと同じであるが，LEAでは上記のネットを水平トラックに割当てる時”左から順”に割当てるのに対して，このアルゴリズムでは次の問題で述べられるような意味で最適に割当てている。

問題：垂直制約グラフ上で先祖を持たないネットの集合をNとし，各ネット$n \in N$に対して重み関数$F(n)$が与えられているものとする。この時，互いに水平線分の重ならないネットの集合のうち

$$\sum_{n \in N_a} F(n) \rightarrow \max$$

とする$N_a \subseteq N$を求めよ。

[例]ネット集合をN={1, 2, 3, 4, 5, 6}，各ネットのゾーン表現を図2.22とし，重み関数を$F(1)$=3，$F(2)$=3，$F(3)$=2，$F(4)$=2，$F(5)$=2，$F(6)$=1とする。このとき

$$N_a = \{2, 4, 5\}$$

$$\sum_{n \in N_a} F(n) = 7$$

である。

ここで，ゾーンとネットの左端，右端の関係を表わす有向グラフ$G=(N, E)$を考える。ただし，Nはノードの集合であり，各ノードはゾーン番号(便宜上ゾーン0を含む)と1対1に対応する。またEは次のように設定される有向枝の集合である。

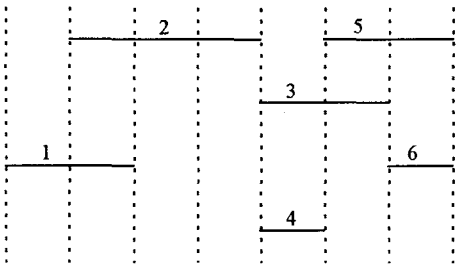


図 2.22　ゾーン表現

- (各ゾーン Z 毎に)ゾーン Z-1 に対応するノードからゾーン Z に対応するノード　間に向う枝を挿入し，その重みを 0 とする
- (各ネット N 毎に)ネット N の左端ゾーンを Z_L 右端ゾーンを Z_R とする時，ゾーン　Z_L-1 に対応するノードからゾーン Z_R に対応するノード間に向う枝を挿入し，その重みをネット N の重みとする。

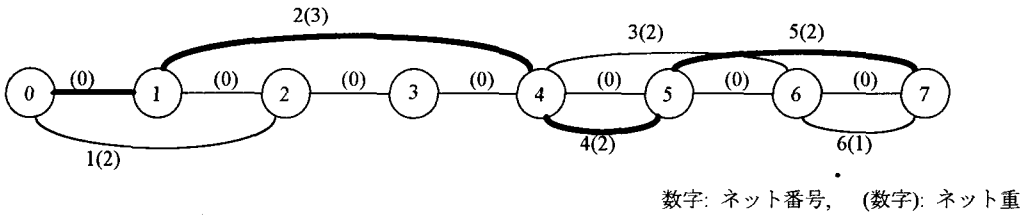


図 2.23　最適ネット割り当てを計算するためのグラフ

前記の例に対応するグラフを図 2.23 に示す。明らかに，このグラフ上での最長経路(経路中のネットの重みの和が最大)を計算することにより N_a が求まる。 しかし，この問題に対しては，実際に上記のグラフを作成する必要はなく，ネットの開始ゾーン，終了ゾーンを格納するリストから直接 N_a を求めることができる。 以下そのアルゴリズムを提案する。

アルゴリズム

// $z_s(n)$: net N の開始ゾーン

// $z_e(n)$: net N の終了ゾーン

// $f(n)$: net N の重み関数

// $z_n(z)$: ゾーン Z で終了するネットの集合

// $z_p(z)$: ゾーン Z の potential

// $z_a(z)$: ネット N_a の集合を表現するリスト構造

// #z　: ゾーン数

$Z_p(0)=0$　;

$Z_a(0)=0$　;

```
for i=1 to #z do;
    Zp(i)=Zp(i-1);
    Za(i)=Zn(i-1);
    for n∈Zn(i) do;
        p=Zp(Zs(n)-1)+F(n);
        if p≧Zp(i)
            then
                Zp(i)=p ;
                Za(i)=n ;
            end
    end
```

この処理で得られた Z_a から次の手順で N_a を求める。

```
Na={}
n=za(#z)
while n=0 do
    Na=Na ∪ {n};
    n=Za(Zs(n)−1);
end
```

従って，このアルゴリズムを各水平トラックごとに実行することによって一つの解が求まる。一つのトラックにネットを割当てる手間は候補となっているネット数に比例することは明らかである。

つぎの問題は，ネットの重み関数 $f(n)$ の決定である。考慮すべき項目は，各ゾーンの局所配線密度，垂直制約グラフの最長経路長等である。ここでは，次のように定める。まず，各ゾーン毎にそのゾーンの局所配線密度 d_z に基づいて関数 $g(d_z)$ を次のように定義する。

$g(d_z)$ = 10 : max density−localdensity = 0 のとき
5 : 〃 = 1 〃
3 : 〃 = 2 〃
2 : 〃 = 3 〃
1 : 〃 = 4 以上のとき

これに対して，ネット n の重み $f(n)$ を次式で定める。

$f(n)$=n の通過するゾーン毎の $g(d_z)$ の和
+(垂直制約グラフにおいて n を通る最長経路長 × α
+ 垂直制約グラフにおいて n につながる枝数) × n の通過するゾーン数

ここで，αは入力パラメータであるが，実験ではこの値を変化させても解はほとんど変化しなかった。

次に述べる実験に使ったプログラムではパラメータαの値を3としている。また水平トラックの処理順序は上から下へ順に処理するのではなく，まず上端のトラック，次に下端のトラック，その次は上から2番目という様に上下交互に処理している。

2.6 実験結果と考察

本文で提案したアルゴリズムに基づくプログラムをFORTRANで作成し，ACOS700上で実行した。

(1)使用プログラム

以下のプログラムを作成し，評価した。

①プログラム#1: アルゴリズム#1に基づくプログラム。

②プログラム#2: アルゴリズム#2に基づくプログラム。

両プログラムとも開始列を表すパラメータを持っている。これまで，チャンネル配線時にゾーン1からゾーンn(nは最大ゾーン数)まで順に処理をすると述べた。しかし，必ずしもこの順に処理する必要はない。実際，局所配線密度が最大となるゾーンから処理した方が良い結果が得られた。上記のパラメータは処理開始ゾーンを指定するものであループログラムは指定されたゾーンからゾーンnに向って処理を行ない，次にそのゾーンからゾーン1に向って処理を行なう。

③プログラム#3: 高速算法に基づくプログラム

なお，比較のため文献2のプログラム(LTX)もプログラムLとして引用する。

(2)使用データ

例題1～6は既存の論文[3]より取った。また'Difficult Example'はBell研究所のDeutschとSchweikertより入手した。

(3)実験結果

ドッグレッグなし問題:

ドッグレッグを禁止して実行した。表3に最適解，プログラムL，Algorithm#1，Algorithm#2のそれぞれの必要トラック数を示した。また，Algorithm#1とAlgorithm#2の処理時間もあわせて示した。この結果によると，Algorithm#1とAlgorithm#2はいずれも例1～例5に対して最適解を求めていることがわかる。また，すべての例題についてleft edge algorithmより大幅に良い解を得ている。

表 2.3 実験結果(ドッグレッグなしの場合)

例題	最適解	Algorithm#1	Algorithm#2	Algorithm#3	Algorithm#L
例 1	12	12(0.03)	12(0.04)	12	14
例 2	15	15(0.04)	15(0.25)	15	18
例 3	17	17(0.10)	17(0.33)	17	20
例 4	18	18(0.10)	18(0.41)	18	19
例 5	17	17(0.13)	20(0.51)	20	23
例 6	20	20(0.13)	20(0.11)	20	22
Diff.Ex	26*	30(0.23)	28(0.92)	30	39

*Branch & Bound 法による 4 時間の計算で得られた最良解

括弧内は計算時間(単位: 秒)

ドッグレッグ問題:

例 1～例 6 に対してはドッグレッグなしでも最適解が求まっているためここでは difficult example 対して実行した。表 2.4 にその結果を示す。最適解にきわめて近い解が得られていることがわかる。また，algorithm#1 と algorithm#2 の解のドッグレッグの数はそれぞれ 19 個と 18 個であり，Bell 研究所の LTX システムの配線プログラムのドッグレッグ数 50 以上に比べ大幅に少ないことがわかる。図 2.24 に algorithm#2 の結果を示す。

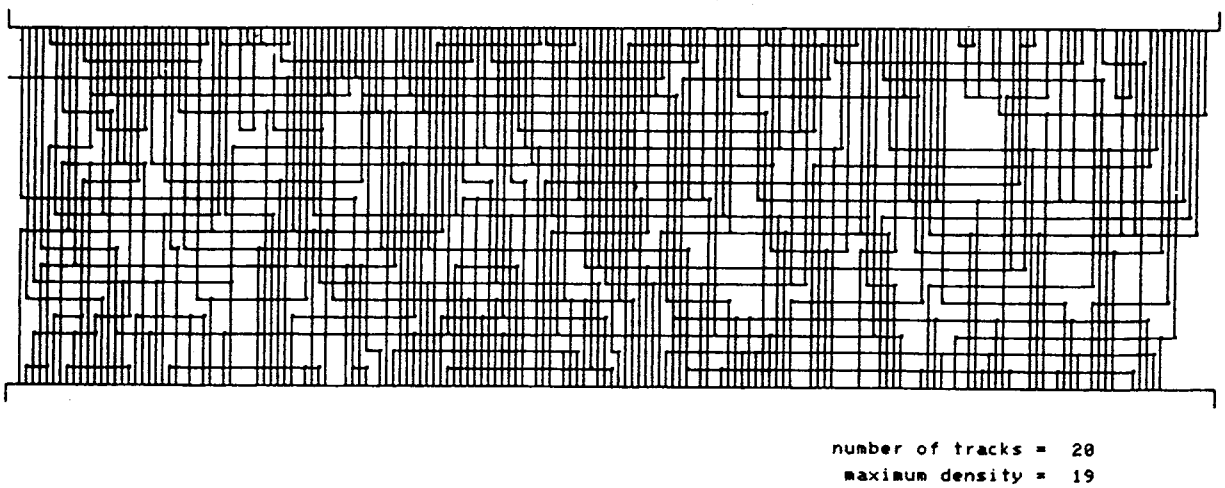


図 2.24 アルゴリズム#2 による Difficult Example の解

表 2.4 実験結果(ドッグレッグありの場合)

例題	配線密度	Algorithm#1	Algorithm#2	Algorithm#3	Algorithm#L
diff.ex	19	21(0.58)	20(1.21)	20(0.59)	21

2.7 結言

本章では，レイアウト設計において重要なチャンネル配線問題を取り上げ，各ネットの配線位置に関する制約を表すグラフ上でネットの併合を繰り返して解く 2 つの手法を提案した。計算機実験によりこの手法が短時間にしかもほとんどの場合最適解を発見し，従来の手法に比べ計算時間，得られる解の良さの点で優れていることを示した。

また，同じチャンネル配線問題に対して，Left Edge Algorithm を改良し，最長経路アルゴリズムを用いて各水平トラックごとの最適な割り当てを繰り返すことによってチャンネル全体の配線行なうアルゴリズムを提案した。この手法は上記の手法と全く異なるが，必要配線領域がほとんど同じ解をさらに高速に計算することができ，しかも水平トラックを 1 本づつ順に処理する方式を用いているため，凹凸のあるチャンネルの処理にも容易に適用できるため，実用上有用である。

3章 コンパクション

本章ではLSIのレイアウトCADにおけるきわめて重要な問題の一つであるコンパクション問題を考察する。この問題に関して，従来より数多くの研究がなされてきたが，それらのほとんどはレイアウト領域の最小化のみを考慮していた。実際の設計においては，面積だけでなく配線長を最小化することも重要であるが，この問題に対して最適解を効率良く求める手法は知られていなかった。ここではまず，この問題が線形計画問題として定式化でき，グラフ上で基本カットセット，基本ループに関する2つの性質を満たす木の発見問題に帰着されることを証明する。次にこの木を木の初等変換の繰り返しにより求める方法を提案し，この手法が線形計画問題の代表的解法であるシンプレックス法に対応することを示す。さらに，上記の総配線長最小コンパクション問題を最小コストフローアルゴリズムを用いて解く方法を提案し，その手法の正当性を証明する。そしてコンパクションに関するこれらの解法が線形計画問題をシンプレックス法を用いてそのまま解くのに比べはるかに効率が良いことを示す。

3.1 緒言

VLSIのCAD，電気回路網，通信網，交通網，水道網における網の解析および設計において各種のネットワーク問題を取り扱う場合が多いが，対象とするシステムの大規模化に伴い，それぞれの問題をいかにして効率よく解くかが重要となっている。そのための一つのアプローチとして，制約条件，目的関数を線形化することによって，線形計画問題として定式化し，シンプレックス法を用いて解く方法がある。一般的には，シンプレックス法は行列演算を伴うため，対象とするシステムが大規模となった場合効率の点で問題が残る。ところが，ネットワークを扱う線形計画問題では，制約条件式の係数行列はネットワークの構造に密接に関連するため完全単模(Totally Unimodular)となることが多い。この場合，基底変換を何回行っても，係数行列に±1，0要素しか現われないという特徴を持つため，グラフ理論と対応づけられた効率のよい解法が存在する。例えば，最小コストフロー問題については，シンプレックス法に直接対応するグラフ理論的解法が広く用いられている。また，最長経路問題，最大フロー問題を始めとするパス，カットセットの最大，最小問題についても，シンプレックス法のグラフ理論的考察がなされている[8]。しかし現実の問題に対するグラフ理論的解法についてはあまり報告がない。本文では，LSIレイアウトコンパクション問題を例として取上げ，線形計画法の立場からそれを解くための新しいアルゴリズムを提案する。

VLSIレイアウトコンパクション問題は，実現可能な初期レイアウトが与えられた時，無駄な領域を削除することにより，全体のレイアウトの最適化を行う問題である。この問題はVLSI・CADにおいて極めて重要な問題であるためこれまで多くの研究がなされているが[9–19]，いずれもレイアウトの領域を最小化することを目的としており，面積の最小化以外の目的関数を考慮したもの[20]は少ない。しかし，実際の設計においては面積だけでなく電気的特性の制約から配線長を最小化することも重要である。そこで，チップ面積を最小化するだけでなく，総配線長を最小化するコンパクションについて考え，この問題が，あるグラフ上で基本ループ，基本カットセット

の関する条件をみたす木を求める問題に帰着されることを示す。そして，それを解くための，シンプレックス法に対応するグラフ理論的アルゴリズムを提案する。これは，行列を扱う代わりに，木の初等変換による解法であるため極めて効率の良い方法である。

また，この手法は他のネットワーク問題，特に最長経路アルゴリズムを用いて解かれている問題に対して応用することが期待出来る。例えば，VLSI コンパクション問題はその典型的な例である。また，PERT，上水道網におけるポンプの圧力決定問題，電気回路の最大遅延時間を決定する問題もその例である。本文では，これらの問題に対する，シンプレックス法のグラフ理論的解法についても述べる。

3.2 問題の定式化

(1)コンパクション問題

初期レイアウトとしてブロックの配置およびそれらの端子間の配線が与えられたとき，各ブロックおよび各配線の水平線分，垂直線分の間の相対的な位置関係を保ち，かつ，それらの各要素間の最小許容間隔の条件のもとで総配線長を最小化する問題がコンパクション問題である。

総配線長を最小化するには，縦方向および横方向の長さを縮小する必要があるが，両方向を同時に考慮するのは難しい。そのため，一方向の最小化を交互に繰り返して行なうものとし，従って以下では縦方向の(1 次元)コンパクションのみを考える。すなわち，レイアウト領域の左下端を原点とする X-Y 座標系において Y 方向のコンパクションだけを考える。この時，問題は次のように定式化される。

(2)定式化

垂直方向のコンパクションを行う場合，Y 座標を決定すべき要素はブロック上辺，ブロック下辺および水平線分である。そこで，これらをレイアウト要素と呼び，その集合を V で表わす。また，初期レイアウトにおいて，各レイアウト要素 $i \in V$ を X 軸上に射影した区間を I_i とする。さらに，レイアウト要素 i の Y 座標の初期値を s_i とし，求めるべき Y 座標を y_i とおく。ただし，配線の水平線分の位置は中心線のそれを表わすものとする。

この時，制約条件はまず各要素の上下関係の保存の制約から

$$y_i - y_j \geq d_{ij}, \quad \forall (i,j) \in E_1$$

ただし E_1 は

$$E_1 = \{(i,j) | I_i \cap I_j \neq \phi, \quad i,j \in V, \quad s_i \geq s_j\}$$

であり，d_{ij} は次に定義する定数である(図 3.1 参照)。

$$d_{ij} = \begin{cases} s_i - s_j: i, j\text{はそれぞれ同一ブロックの上辺，下辺 (図3.1(}a\text{)参照)} \\ s_i - s_j: i\text{はブロック上辺，}j\text{はブロック}i\text{に直接つながる水平線分(図3.1(}b\text{)参照)} \\ s_i - s_j: j\text{はブロック下辺，}i\text{はブロック}j\text{に直接つながる水平線分(図3.1(}c\text{)参照)} \\ 1 \quad : \text{その他 (ただし，単位値1は最小許容間隔)} \end{cases}$$

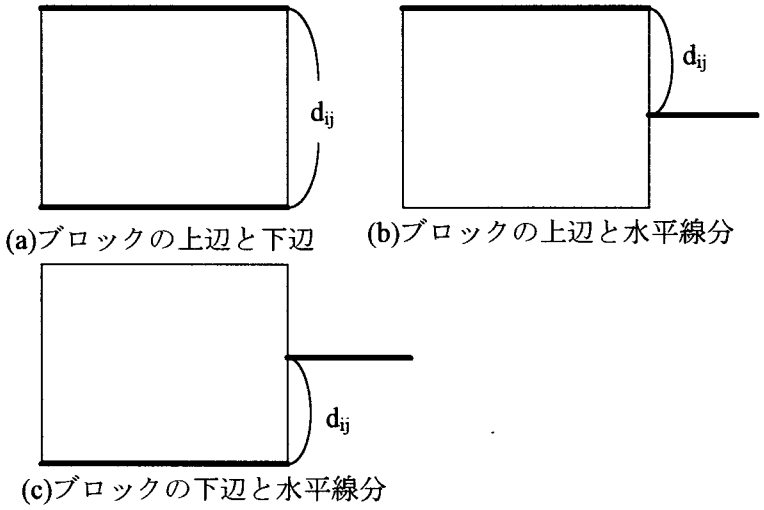


図 3.1 **dij** の説明

次に，ブロックの高さは固定であるから $d_{ji} \equiv -d_{ij}$ とすると

$$y_i - y_j \geq d_{ji}, \quad \forall (i,j) \in E_2$$

ただし，E_2 は

$$E_2 = \{(i,j) | i, j\text{はそれぞれ同一ブロックの上辺，下辺}\}$$

である。また，レイアウト要素 i, j ($y_i \geqq y_j$)の間に垂直線分が存在する時，その長さ ℓ_{ij} は

$$\ell_{ij} = y_i - y_j$$

となるため，c_{ij} を i, j 間の垂直線分数とすると，総配線長 L は

$$L = \sum_{(i,j) \in S} c_{ij} \ell_{ij}$$

$$= \sum_{(i,j) \in S} (y_i - y_j)$$

となる。ここで S は初期レイアウトにおいて $s_i \geq s_j$ となっているレイアウト要素 i, j の集合，すなわち

$$S = \{(i,j) | i, j \in V, s_i \geq s_j\}$$

である。ところで，定義より明らかに

$$S \subseteq E_1$$

であるが，その差集合を

$$S^* = S - E_1$$

とすると，任意の $(i,\ j) \in S^*$ に対して

$$I(i) \cap I(j) \in \phi$$

となる。そのため，レイアウト要素 i，j 間に垂直線分は存在せず，c_{ij} の値は零である。また，明らかに，E_2 の各要素(i,j)に対しても c_{ij} の値は 0 となる。従って，総配線長 L は

$$E = E_1 \cup E_2$$

とすると

$$L = \sum_{(i,j)\in E} c_{ij}(y_i - y_j)$$

となる。

(3)垂直制約グラフ

各レイアウト要素間の相対位置の制約を表現するため，前節で定義した集合 V をノード集合，E を枝集合とする有向グラフ $G=(V,\ E)$を導入し，垂直制約グラフと呼ぶ。垂直制約グラフに基づき，制約条件，目的関数を整理すると次のようになる。

$$y_i - y_j \geq d_{ij}, \quad \forall (i,j) \in E \tag{3.1}$$

$$L = \sum_{(i,j)\in E} c_{ij}(y_i - y_j) \rightarrow \text{最小} \tag{3.2}$$

(3.1)式は，枝 $(i,\ j) \in E$ の長さを d_{ij} と定義するとノード i がノード j よりその長さ d_{ij} 以上だけ上になければならないことを表わしている。なお，間隔の固定されたノード対の場合は，互いに向きが逆で長さの和が 0 となる 2 本の枝の対で表現される。例えば，図 3.3 は図 3.2 の初期配置の例に対応する垂直制約グラフを表わしている。この例ではブロック B_1 の上辺と下辺の距離は固定さ

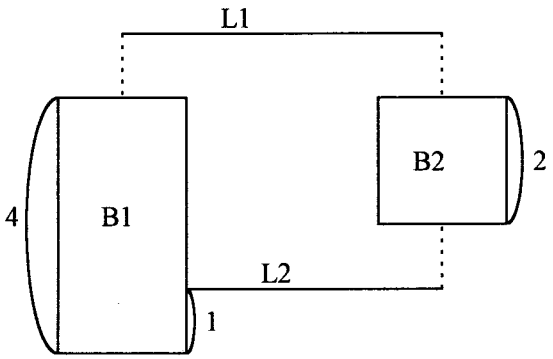


図 3.2 初期レイアウトの例

れているので，互いに逆向きの二本の枝が設定されている。

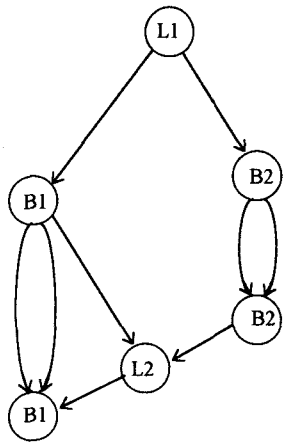


図 3.3　図 3.2 のレイアウトに対応する
垂直制約グラフ

初期レイアウトが実現可能な時，垂直制約グラフ G 上に枝の長さの和が非負となる有向ループは存在しないことは明らかである。従って，このグラフ上で最長経路を計算することによって高さを最小とするレイアウトを求めることができる。

本文では配線長を最小化する問題を扱うため，以下では垂直制約グラフの各枝に長さの以外の値を定義することによって解を求めることとする。なお，各レイアウト要素(ブロック上辺，ブロック下辺，水平線分)は垂直制約グラフのノードに 1 対 1 に対応するので，以下，誤解を生じるおそれのない限り，レイアウト要素と垂直制約グラフのノードを同一視する。

3.3 グラフ理論的考察

スラック変数 $W(\geqq 0)$を導入すると制約条件は次のようになる。

$$y_i - y_j - w_{ij} = d_{ij}\ , \quad \forall (i,j) \in E \tag{3.3}$$

また，目的関数は

$$L = \sum_{(i,j)\in E} c_{ij}(y_i - y_j)$$

$$= \sum_{(i,j)\in E} c_{ij} w_{ij} + D \tag{3.4}$$

となる。ただし，D は次式で与えられる定数である。

$$D = \sum_{(i,j)\in E} c_{ij} d_{ij}$$

ところで，A を垂直制約グラフ G の接続行列とすると，(3.3)式は両辺の符号を反転して次のように行列表現することができる。

$$\begin{bmatrix} 1 & \vdots & -A^T \end{bmatrix} \begin{bmatrix} w \\ \cdots \\ y \end{bmatrix} = \begin{bmatrix} -d \end{bmatrix} \tag{3.5}$$

ここで，垂直制約グラフ G=(V, E)に仮想のノード v，および各ノードからその仮想のノード v へ向かう枝の集合 E'を追加したグラフ G'=($V \cup \{v\}$, $E \cup E'$)を考える.このとき，E'はグラフ G'上の木を構成し(3.5)式の行列$[1: -A^T]$はその木に関する基本ループ行列となっていることがわかる。また，この行列の変数 w に対応する列は E の枝に，変数 y に対応する列は E'の枝に対応していることがわかる。そこで，グラフ G'の枝 $(i, j) \in E$ の長さおよびコストをそれぞれ d_{ij}, c_{ij} とし，枝 $(i, z) \in E'$ の長さとコストのいずれをも 0 と定める。図 3.3 のグラフに対応するグラフ G'を図 3.4 に示す。ただし，G'の各枝の重みは順序対(d_{ij}, c_{ij})を表わす。

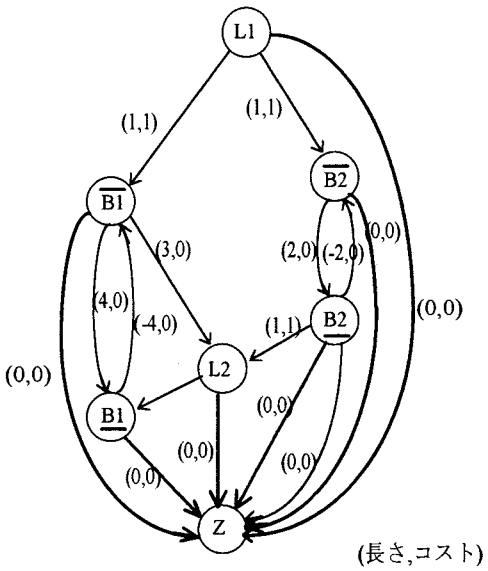


図 3.4　グラフ G'の例

このとき，以下の補題が成立する。

補題 1: (基底解) 任意の基底解の非基底変数に対応する枝はグラフ G 上の木を構成する。

(証明)　垂直制約グラフ G のノード数を n とすると，行列$[1: -A^T]$はグラフ G'におけるランク n の基本ループ行列となっている。従ってグラフ理論の基本定理[1]により任意の独立な n 個の列ベクトルすなわち基底変数に対応する枝は，グラフ G 上のある木に関する補木枝に対応し，それ以外

の列ベクトル，すなわち非基底変数に対応する枝はその木を構成する。(証明終)

また，基本ループの性質から，任意の基底解に対する制約式のシンプレクスタブロー表現はその基底解に対応する木に基づく基本ループ行列となる。

補題 2: (主実行可能条件) ある基底解が主実行可能であるための必要十分条件は，グラフ G'において，その基底解に対応する木に基づく基本ループの値，すなわちそれに含まれる枝の長さの和

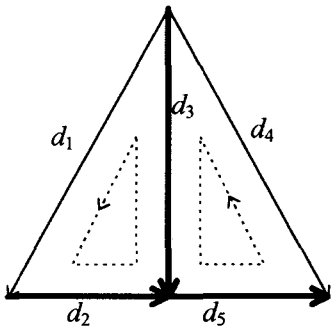


図 3.5　基本ループの値

をその向きに沿って加えた値が全て非正であることである。

図 3.5 を例にとると

$$d_1 + d_2 - d_3 \le 0$$

$$d_4 + d_3 - d_5 \le 0$$

が主実行可能のための条件である。

(証明)ある基底解に対応する制約式のシンプレックスタブロー表現は次のように書ける。

$$\left[B: A^*\right]\begin{bmatrix} u \\ w \end{bmatrix} = -d^*$$

ただし上記の行列における列ベクトルおよび変数の順序は(3.5)式と同じとする。このとき

$$d^* = B\,d$$

となるから，主実行可能条件

$$-d^* \ge 0$$

は，

$$\left[B: A^*\right]\begin{bmatrix} d \\ 0 \end{bmatrix} \le 0$$

と書くことができる。ところで，$[B: A^*]$ は基底解に対応する木に基づく基本ループ行列であり，変数 u，w に対応する枝の長さはそれぞれ 0，d であることから，この式はすべての基本ループの値が非正であることを表わしている。(証明終)

補題 3: 双対実行可能条件ある基底解が双対実行可能であるための必要十分条件は，グラフ G' において，その基底解に対応する木に基づく基本カットセットの値，すなわちそれに含まれる枝のコストをその向きに従って加えた値が全て非負となることである。

図 3.6 を例にとると

$$c_2 - c_1 \geq 0$$
$$c_3 + c_1 - c_4 \geq 0$$
$$c_5 - c_4 \geq 0$$

が双対実行可能のための条件である。

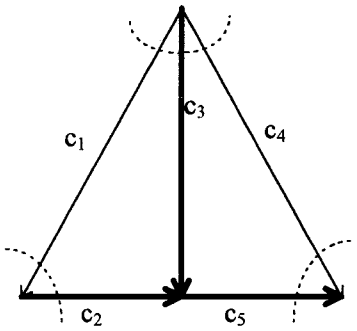


図 3.6　基本カットセットの値

(証明) ある基底解[x_N: x_B]に対する制約式のシンプレクスタブロー表現は次のように書ける。

$$[1:S]\begin{bmatrix} x_B \\ x_N \end{bmatrix} = d^*$$

ここで x_B, x_N をそれぞれ基底変数，非基底変数とし，c_B, c_N をそれぞれ(3.4)式の目的関数における変数 x_B, x_N の係数ベクトルとすると，目的関数は

$$L = c_B x_B + c_N x_N$$
$$= c_B(d^* - Sx_N) + c_N x_N$$
$$= (c_N - c_B S)x_N + c_B d^*$$

となるから，双対実行可能条件は

$$c_N - c_B S \geq 0$$

となる。この式は，また

$$[-S^T:1]\begin{bmatrix} c_B \\ c_N \end{bmatrix} \geq 0$$

と行列表現できる。ところで，行列[1: S]は，上記の基底解に対応する木の基本ループ行列であるため，行列[$-S^T$: 1]は基本カットセット行列となる。従って，上式は基本カットセットの値が非負であることを意味している。(証明終)

従って，コンパクション問題はグラフ G'上で上記の二条件を満たす木(以下，この木を最適木とよぶ)を求める問題となる。初期レイアウトが実行可能，すなわち制約条件を満たしている場合，最適木の存在は保証されるが，より一般的な場合について，最適木の存在に関して次の補題が成立する。

補題 4：主実行可能な解が存在するための必要十分条件は，グラフ G'上に枝の長さの和が正となるサイクルが存在しないことである。

証明：(十分性) グラフ G'上に枝の長さの和が正となるサイクルが存在しないとき，各ノードからノード z に至る最長経路木が存在する。この木に関する基本ループの値はすべて非正となるため，この木に対応する解は主実行可能である。

(必要性) 主実行可能な解，すなわち，基本ループの値がすべて非正となる木が存在すると仮定する。そして，p_i を各ノード i から木にそってノード z に至る距離，すなわち経路上の木枝の長さをその枝の向きに従って加えた(引いた)値とする。各枝(i, j)の長さを d_{ij} とすると，仮定より

$$p_i - p_j \geq d_{ij}$$

である。そこで，枝の長さの和が正となるサイクルがあるとした場合，そのサイクルを (i_1, i_2), (i_2, i_3), ..., (i_m, i_1)とすると

$$p_{i_1} - p_{i_2} \geq d_{i_1 i_2}$$

$$p_{i_2} - p_{i_3} \geq d_{i_2 i_3}$$

.....

$$p_{i_m} - p_{i_1} \geq d_{i_m i_1}$$

となる。これらの式の両辺を足し合わせると

$$0 \geq d_{i_1 i_2} + d_{i_2 i_3} + \cdots + d_{i_m i_1}$$

となるが，右辺の値はサイクル中の枝の長さの和であるため正となり矛盾する。従って，枝の長さの和が正となるサイクルは存在しない。 (証明終)

補題 5：グラフ G'に枝の長さの和が正となるサイクルが存在しないとき，任意の最長経路木に対応する解は主実行可能である。

(証明) 最長経路木に関する基本ループの値はすべて非正となる。従って，補題 2 よりこの木に対応する解は主実行可能である。 (証明終)

3.4 木の初等変換による算法

(1)総配線長の最小化

グラフ G'上で基本ループの値が非正，基本カットセットの値が非負となる木を求めるアルゴリズムを示す。このアルゴリズムは，線形計画法におけるシンプレックス法に基づいたものである。

アルゴリズム:

Step 1 :グラフ G'において最長経路木 T を求める。

Step 2 :T の基本カットセットの値を負とする木枝がある時，以下に述べる木枝 e_t と補木枝 e_c による木の初等変換を行う。ここで

(1)e_t は基本カットセットの値が負となる木枝

(2)e_c は次の 2 条件を満たす補木枝

① e_t の基本カットセットに含まれ，かつ枝の向きがカットセットの向きと逆である。

②上記の条件を満たす補木枝の内，基本ループの値が最大である。

この処理を基本カットセットの値を負とする木枝が無くなるまで繰返す。

Step 3 :各ノードから，木枝を通って仮想ノード z に到る経路長を計算し，その値を対応するレイアウト要素の y 座標とする。

(アルゴリズム終)

Step1 はシンプレックス法の Phase1 に相当し，基本ループの値がすべて非正の木，すなわち主実行可能基底解を求めている。

Step2 はシンプレックス法の Phase2 に相当し，木の初等変換により，基本ループの条件を満たし，かつ基本カットセットの値がすべて非負の木を求めている。なお，手続き中の条件①は基本カットセットの向きを反転して，その値を正にするため，また条件②は木の初等変換後の基本ループの値を非正に保つためのものである。

また，Step3 では非基底変数の値をもとに各レイアウト要素の Y 座標を計算している。

(例) 図 3.7 の初期レイアウトに対応する最長経路木を図 3.8 に示す。ただし，図 3.8 では図を簡潔にするため仮想のノード z およびそれにつながる仮想枝は削除した。なお，仮想枝の長さ，コストはいずれも 0 であるため，実際に上記のアルゴリズムを適用する時は，このグラフで十分である。図 3.8 の木に対応する解は主実行可能である。しかし木枝(e, f)に関する基本カットセットの値が負(1–0–1–1=–1)となるため双対実行可能ではない。そこで，木の初等変換により，この木から枝を取り除き新しい枝を挿入する。その候補としては Step2 の条件①より枝(d, e)，(c, e)と(b, e)があげられる。それらの内，枝(b, e)が最大の基本ループの値(–1)を持つので，Step2 の条件

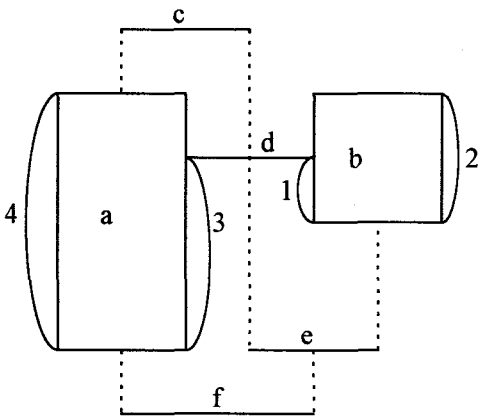


図 3.7 初期レイアウト

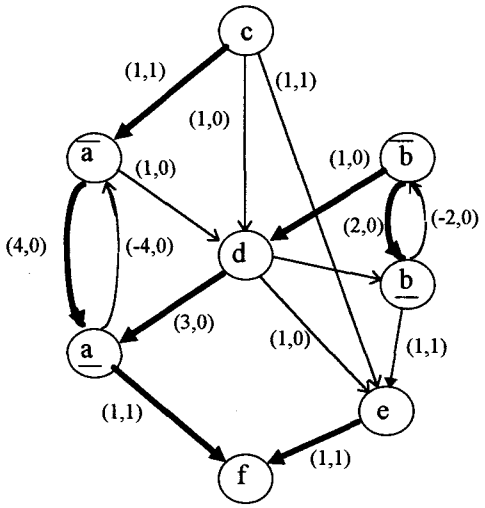


図 3.8 最長経路木[(,)は(長さ,コスト)]

②から枝(b, e)が追加される枝として選ぶ。図 3.9 に木の初等変換により生成された木を示す。この木は図 3.10 の解に対応している。図 3.7 に解に比べ水平線分 e が上方向に移動し，総配線長が減少していることがわかる。

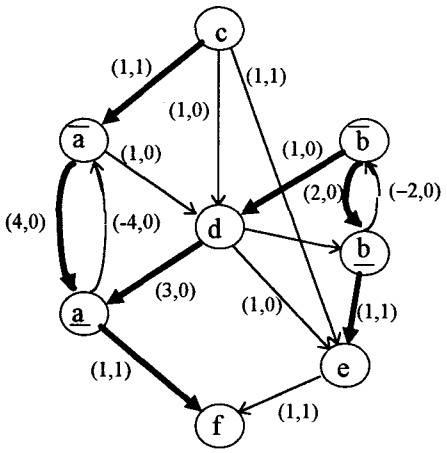


図 3.9 最適木

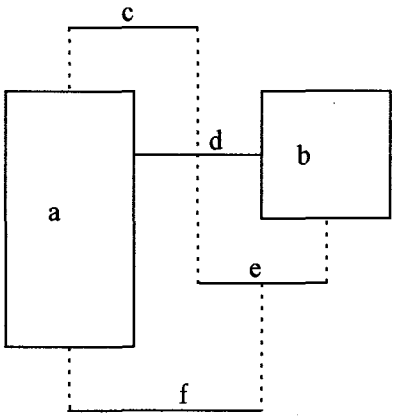


図 3.10 最適解

(2)レイアウト面積の最小化

これまでは総配線長の最小化について述べ，面積の最小化は考慮しなかった。実際，配線長を最小化しても必ずしも面積は最小とならない。例えば図 3.12 のレイアウトは図 3.11 のレイアウトよりも配線長は短くなっているが，逆に全てのレイアウト要素を囲む矩形の面積は大きくなっている。面積を最小化し，それが増加しない範囲内で総配線長を最小化するには，上記のアルゴリズムの Step 1 と Step 2 の間に次の Step を実行すればよい。

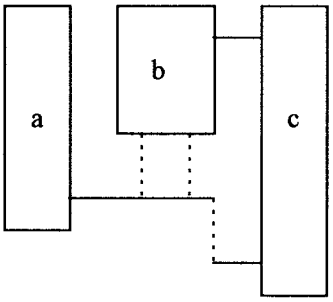


図 3.11 面積最小レイアウト

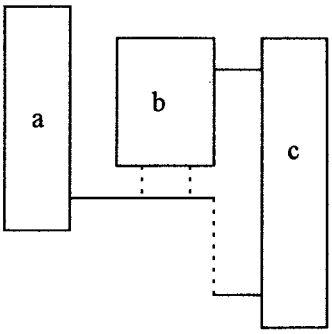


図 3.12 配線長最小レイアウト

Step 1.5:

Spep 1 で得られた最長経路長を L とすると，グラフ G'上で入る枝を持たない各ノードに対して，そのノードを終点，仮想ノード z を始点とする長さ-L の枝を追加する。

これは，式(2.1)の制約条件に，さらに次の制約を追加したことを意味している。

$$y_i - y_z \le L, \quad \forall i \in V - \{z\}$$

この条件を付加しても主実行可能性は明らかに保たれるので，面積(高さ)を最小とし，そのもとで配線長を最小化する解を求めることができる。

3.5 最小コストフローアルゴリズムを用いた算法

前節では，制約グラフが与えられた時，基本ループの値と基本カットセットの値が全て正の木を，木の初等変換により求める方法を提案したが，ここでは，同じ木を最小コストフローアルゴリズムによって求める手法を提案する。そのため，まずグラフ G を部分グラフとして含むグラフ G^*を定義し，グラフ G^*上のフローとグラフ G 上の木を対応させ，両者の関係を調べる。又，グラフ G 上で求める木が存在するための条件を導く。なお，以後とくに断わらない限り，フローはグラフ G^*上で，基本ループ，基本カットセットは，グラフ G 上で定めるものとする。

なお，前節では基本カットセットを“枝のコスト”を用いて定義したが，本節では枝コストと絶対値が同じで符号が異なる“枝の重み”を用いる。従って，基本カットセット条件はすべての基本カットセットの値が非正となることに変わる。枝の重みを使用する理由は，本節では最小コストフローを扱うが，その際フローのコストとの間で混乱を招くことを避けるためである。

(1)諸定義

グラフ $G=(V, E)$における各枝 $e(\in E)$ の長さ，重みをそれぞれ $d(e)$, $w(e)$とかく。また，各ノード $v \in V$ から出る枝の集合を $D^+(v)$，入る枝の集合を $D^-(v)$とし，ノード v に対する重み $x(v)$を次式で定義する。

$$x(v) = \sum_{e \in D^+(v)} w(e) - \sum_{e \in D^-(v)} w(e) \tag{3.6}$$

この値を用いて，グラフ $G^*=(V^* \cup \{s,t\},\ E_s \cup E_t)$ を定義する。

ただし s, t はそれぞれフローの始点，終点を表わし，E_s, E_t は次式で定義される枝の集合である。

$$E_s = \{(s,v) \mid v \in V,\ x(v) > 0\}$$

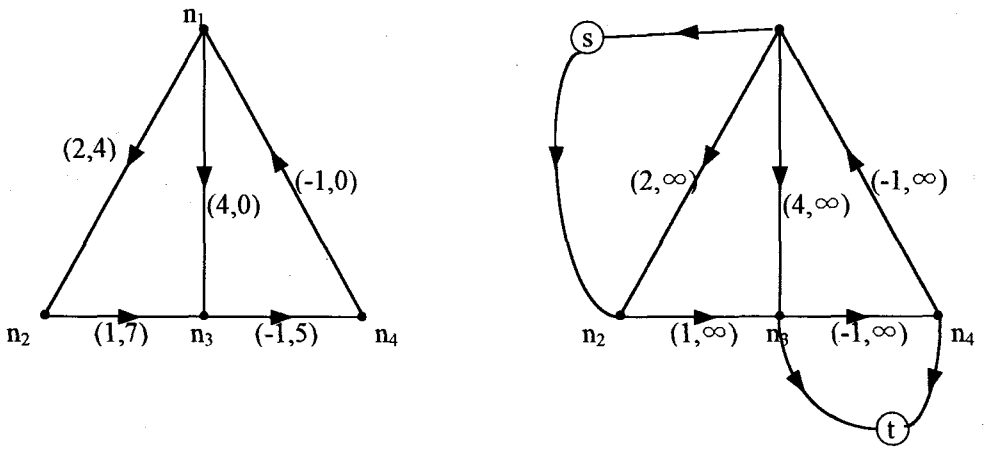
$$E_t = \{(v,t) \mid v \in V,\ x(v) < 0\}$$

さらに，各枝 e の容量 $p(e)$，コスト $c(e)$ を次式で定める。

$$c(e) = \begin{cases} 0 & : e \in E_s \cup E_t \\ d(e) & : e \in E \end{cases}$$

$$p(e) = \begin{cases} \infty & : e \in E \\ x(v) & : e = (s,v) \in E_s \\ -x(v) & : e = (v,t) \in E_t \end{cases} \qquad (3.7)$$

図 3.13 にグラフ G と対応するグラフ G^* を示す。この例ではグラフ G において $x(n_1)$=4, $x(n_2)$=3, $x(n_3)$=−2, $x(n_4)$=−5 となるから，グラフ G^* はグラフ G にノード s, t と枝$(s,\ n_1)$, $(s,\ n_2)$, $(n_3,\ t)$, $(n_4,\ t)$ を追加して得られる。追加された枝の容量は，それぞれ 4, 3, 2, 5 であり，コストはすべて 0 である。他の枝のコストはグラフ G の枝の長さで，その容量は∞である。



(a)グラフ G[数字は(長さ,重み)]　　(b)グラフ G^*[数字は(コスト,容量)]

図 3.13　グラフ G と対応するグラフ G^*

グラフ $G^*=(V^*,\ E^*)$ 上で s から t へのフローを

$$\mathbf{f} = (f_1, f_2, \cdots, f_e)$$

で表す。ここで f_i は枝 $i \in E^*$ のフロー量で，それが枝の向きに流れるときを正，逆向きのときを負とする。また，グラフ G^* の各枝のコストを

$$\mathbf{c} = (c_1, c_2, \cdots, c_E)$$

とするとき，フロー $\mathbf{f}$ のコストを

$$\mathbf{c}\cdot\mathbf{f}=\sum_{i\in E^*}c_i f_i$$

で定義し，s から t への最大フローの内，コストの最も小さいものを“コスト最小の最大フロー”と呼ぶ。また，グラフ G^*の s から t へのフローを **f** とするとき，枝 E の内フロー量が非零の枝の集合を E_f とする。

(2)フローの性質

最小コストの最大フローに関して，明らかに次の補題が成立する。

補題 1: グラフ G^*における s から t への任意のフローを **f** とすると，$E_s \cup E_t$ の各枝のフロー量が **f** と同じで，しかも E_fの枝だけからなるループを含まないフロー**f'**が存在する。特に，**f** がコスト最小の最大フローのときは，フローのコストが **f** と同じで，かつ上記の条件を満たすフロー**f'**が存在する。

そのため，以後，本文で扱かうコスト最小の最大フローは，グラフ G*上で E_fの枝だけから成るループを含まないものとする。

(3)最大フローと基本カットセットの値

フローと木の間に次の関係がある。

補題 2: グラフ G^*において $E_s \cup E_t$ の枝を全て飽和させる s から t へのフロー**f** が存在するとき，E_fの全ての枝を含むグラフ G 上の任意の木 T に対して，Tの各枝の流量はその枝に関する基本カットセットの値に等しい。

証明: グラフ G において，Tの任意の枝を e_t=(v_1，v_2)とし，e_tによって分割される T 上のノード V の集合を V_1，V_2(ただし $v_1 \in V_1$，$v_2 \in V_2$)とする。また，e_tの基本カットセットを E^+，E^-とする。ただし，E^+は e_t と同じ向きの枝，E^-は e_t と逆向きの枝の集合である。

また，グラフ G^*においてsから V_1 につながる枝の集合を E_{s1}，その容量の和を $P(E_{s1})$とする。同様に $P(E_{s2})$，$P(E_{t1})$，$P(E_{t2})$を定義する(図 3.14)。

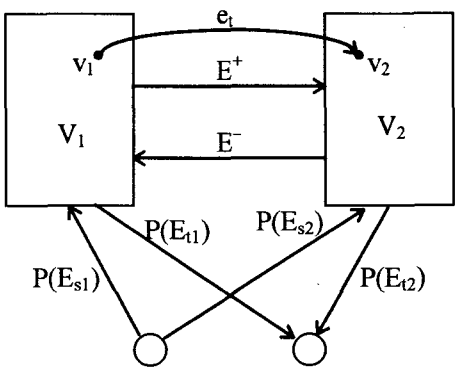


図 3.14 $P(E_{s1})$, $P(E_{s2})$, $P(E_{t1})$, $P(E_{s2})$,の定義

このとき，(3.7)より

$$\sum_{v\in V_1} x(v) = P(E_{s1}) - p(E_{t1}) \tag{3.8}$$

(3.6)より

$$\sum_{x\in V_1} x(v) = \sum_{x\in V_1}\{\sum_{e\in D^+(v)} w(e) - \sum_{e\in D^-(v)} w(e)\}$$

$$= \sum_{e\in E^+\cup\{e_t\}} w(e) - \sum_{e\in E^-} w(e) \tag{3.9}$$

(3.8)，(3.9)より

$$P(E_{s1}) - P(E_{t1}) = \sum_{e\in E^+\cup\{e_t\}} w(e) - \sum_{e\in E^-} w(e) \tag{3.10}$$

一方，E_s，E_t の枝がすべて飽和し，E^+，E^-の枝の流量が 0 であることから，枝 e_tのフロー量$f(e_t)$は流量の保存則より

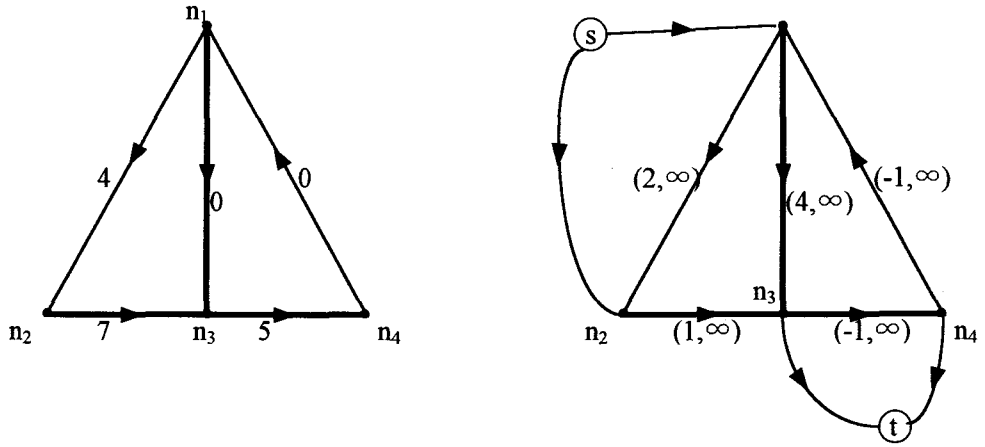
$$f(e_t) = P(E_{s1}) - P(E_{t1}) \tag{3.11}$$

(3.10)，(3.11)より

$$f(e_t) = \sum_{e\in E^+\cup\{e_t\}} w(e) - \sum_{e\in E^-} w(e)$$

となり，これは枝 e_tのカットセットの値に等しい。

図 3.15(b)は s，t 間のフローの例，図 3.15(a)はこのフローに対応するグラフ G の木を表す。 枝 e_2，e_3，e_5のカットセットはそれぞれ 3，4，5 で，これらは各枝のフロー量に等しいことがわかる。補題 2 で，f の各枝の流量が全て正であることに注意すれば，T の基本カットセットは全て正であることがわかる。



(a)グラフ G[数字は(長さ,重み)]　　(b)グラフ G^*[数字は(コスト,容量)]

図 3.15 グラフ G と対応するグラフ G^*

(4)基本カットセットの値が全て非負の木の存在

補題2では，E_s，E_t を飽和させるフローが存在する場合を扱かったが，ここでは，グラフ G^* にこのようなフローが存在するための条件およびグラフ G に基本カットセットの値が全て非負の木が存在するための条件について述べる。

補題 3: グラフ G において基本カットセットの値が全て非負の木が存在するための必要十分条件は，グラフ G^* において E_s，E_t の枝を全て飽和させる s から t へのフローが存在することである。

証明: (十分性)グラフ G に，基本カットセットの値が全て非負の木 T があるとき，グラフ G^* で各枝の流量 f(e)が次式で与えられる流量は容量制限，流量保存則をみたす。

$$f(e)=\begin{cases} \text{枝eの基本カットセットの値} & :e\in T \\ 0 & :e\in E-T \\ p(e) & :e\in E_s\cup E_u \end{cases}$$

(必要性)補題2より明らか。

最大フロー最小カット定理から次の補題が得られる。

補題 4: グラフ G^* において E_s，E_t の全ての枝を飽和させる s から t へのフローが存在するための必要十分条件は，グラフ G に，枝の重みの和が負で，向きのそろったカットセットが存在しないことである。

証明: (十分性)グラフ G に，枝の重みの和が負でしかも向きの揃ったカットセットCが存在するとき，Cによって分けられるノードの集合を V_1，V_2 とし，Cの向きを V_1 から V_2 とする。グラフ G^* においてsから V_1 に向かう枝の容量の和を $P(E_{s1})$とし，同様に，$P(E_{s2})$，$P(E_{t1})$，$P(E_{t2})$を定義する。このときグラフ G^* の定義より

$$P(E_{s1})+P(E_{s2})=P(E_{t1})+P(E_{t2})$$

(3.7)より

$$P(E_{s1})-P(E_{t1})=\sum_{v\in V_1}x(v) \tag{3.12}$$

(3.6)より

$$\sum_{v\in V_1}x(v)=\sum_{e\in C}w(e) \tag{3.13}$$

Cの枝の重みの和は仮定より負であるから，(3.12)，(3.13)より

$$P(E_{s1})-P(E_{t1})=\sum_{e\in C}w(e)<0 \tag{3.14}$$

となる。ここで(3.13)，(3.14)より

$$P(E_{t1})+P(E_{t2})>P(E_{s1})+P(E_{t2})$$

となり，グラフ G^*においてカットセット C*を

$$C^* = C \cup E_{S1} \cup E_{t2}$$

と定めると(図 3.14)，C*の枝 s から t へ向かう容量の和は $P(E_{s1})$+$P(E_{t2})$であるから，最大フロー最小カット定理により，E_{t1}，E_{t2} を全て飽和させる s から t へのフローは存在しない。

(必要性)グラフ G^*において E_s，E_t のすべての枝を飽和させる s から t へのフローが存在しないとする。この時，最大フロー最小カット定理により，グラフ G^*に次の性質を持つカットセット C^* が存在する。

性質 1： C^*は s と t を分離する。

性質 2： C^*の s から t へ向かう容量の和は E_t の枝の容量の和より小さい(E_s の枝の容量の和より小さい)。

性質 2 より C^*は E_s の枝だけからなることはなく，E_t の枝だけからなることもない。さらに，G が連結であることから $E_s \cup E_t$ の枝だけからなることもない。従って C を

$$C = C^* \cap E$$

とすると C はグラフ G のカットセットになり，さらに C の各枝の容量が∞であることに注意すれば性質 1 より C はすべて t から s の向きである。そこで，C によって分割される V のノードの集合を V_1，V_2(C の向きを V_2 から V_1 とする)とし，s から V_1 に向かう枝の容量の和を $P(E_{s1})$，同様に $P(E_{s2})$，$P(E_{t1})$，$P(E_{t2})$を定義したとき

$$P(E_{s2})+P(E_{t1})<P(E_{s1})+P(E_{s2}) \tag{3.15}$$

となる。(3.15)より

$$P(E_{s1})-P(E_{t1})>0 \tag{3.16}$$

一方，カットセット C の各枝の重みは

$$\sum_{e\in C} w(e) = -\sum_{x\in v_1} x(v)$$

$$= P(E_{t1})-P(E_{s1}) \tag{3.17}$$

であり，(3.16)，(3.17)より

$$\sum_{e\in C} w(e) < 0$$

となるから，グラフ G に枝の重みの和が負で向きのそろったカットセット C が存在することがわかる。

補題 3，補題 4 より次の補題が得られる。

補題 5: グラフ G において基本カットセットの値が全て非負の木が存在するための必要十分条件は，グラフ G に，枝の重みの和が負で向きのそろったカットセットが存在しないことである。

なお，現実の問題では枝の重みは全て非負であり，またグラフの構造も制約されているので，基本カットセットの値が全て非負の木は必ず存在する。

(5)基本ループの値が全て正の木の存在

補題 6:グラフ G において基本ループの値が全て非負の木が存在するための必要十分条件は，グラフ G に，枝の長さの和が負で，枝の向きのそろった有向ループが存在しないことである。

(6)コスト最小の最大フロー

グラフ G に，枝の長さの和が負で枝の向きのそろった有向ループが存在しないとき，グラフ G^* に s から t へのコスト最小の最大フロー**f** が存在し，それに対応する E_f は次の性質を有する。

先ず，用語の定義を行う

グラフ G の**有向ループの値**とは，その中に含まれる枝の長さをそのループの向きに沿って加えた値とする。例えば図 3.16 の有向ループの値は

$$(-1)+2-(-3)+4-(-1)-3=6$$

である。

なお，以下の議論では，全部あるいは特定の枝の向きがそろった有向ループを考えている。そこで，特に断わらないかぎり，有向ループの向きはこれらの枝の向きにとる。

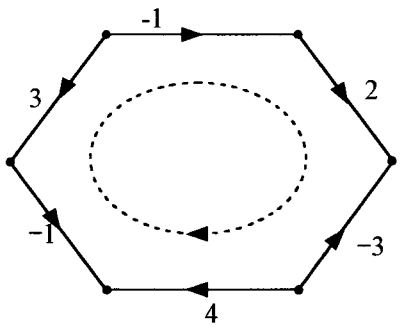


図 3.16 有向ループの値

補題 7 : f をグラフ G^*のコスト最小の最大フローとするとき，グラフ G で E-E_f 枝の向きのそろった有向ループの値は全て非負である。

証明： E の枝のコスト(グラフ G^*)と長さ(グラフ G)は等しいので，グラフ G に，$E-E_f$ の枝の向きがそろい，値が負の有向ループがあれば，グラフ G^*で，ループに沿ったフローをその向きに流すことによって，容量制限を満たしながらフローのコストを下げることが出来る。ところが f はコスト最小のフローであるから，このような有向ループは存在しない。

(7)木の構成

補題2および補題7から，グラフ G^* 上で E_s, E_t の枝を全て飽和させるコスト最小の最大フローfを求めたとき，E_fがグラフ G の木になっていれば，その基本ループの値，基本カットセットの値は全て非負である。E_fが木にならない場合に基本ループの値に注意しながら E_fに枝を追加し，グラフ G 上の木を構成する方法について述べる

まず，”**枝の短絡除去**”という操作を定義する(図3.17)。すなわち，グラフ G の枝 $e=(a, b)$ を短絡除去するとは

(1) グラフ G から枝 e を取り除く。

(2) ノード a と b を縮約して新しい一つのノードとする。

(3) ノード b から出る枝の集合を D^+，入る枝の集合を D^-とすると，各枝 e'の長さ $d(e')$を次のように更新する。

$$d(e') \leftarrow \begin{cases} d(e') + d(e): & e' \in D^+ \\ d(e') - d(e): & e' \in D^- \end{cases}$$

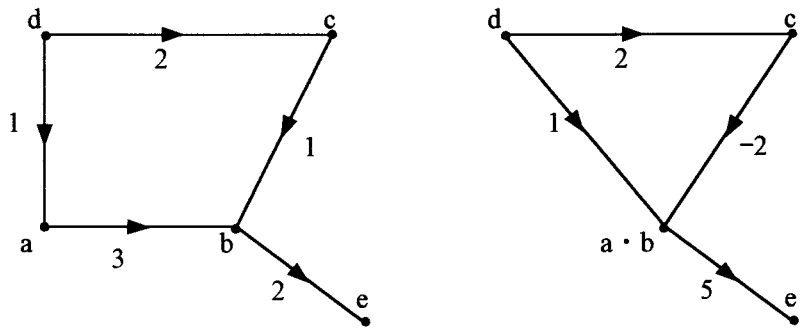


図 3.17 枝(a,b)の短絡除去

グラフ $G=(V, E)$から E_fの枝を任意の順に1本ずつすべて短絡除去したグラフ $G'=(V', E')$, $E' = E - E_f$を考える。グラフ G は E_fの枝だけからなるループを含まないため，グラフ G'の定義から明らかなように，グラフ G とグラフ G'の任意の有向ループの間には1対1の対応関係があり，対応するループの値は等しい。従って，補題7からグラフ G'では枝の向きがそろった有向ループの値は全て非負である。

そこでグラフ G'上で，任意のノード r から到達可能な全ノード集合を $V''\subseteq V'$，枝の集合 E'' を

$$E'' = \{(a,b) | a,b \in V'', (a,b) \in E'\}$$

とするとグラフ $G'' = (V'', E'')$には r をルートとする最短木 T''が存在し，次の補題が成り立つ。

補題8： グラフ G'では $E'-T''$の枝の向きの全てそろった有向ループの値は全て非負である.

証明： V'' の定義から，グラフ G'において V''から $V'-V''$へ向かう枝は存在せず，対象となる有向ループは次の2種類だけである。

(1) $E'-E''$の枝だけからなり，枝の向きのすべてそろった有向ループ:

グラフ G'では，枝の向きのすべてそろった有向ループの値はすべて非負であることから，このループの値も非負である。

(2) $E"$ の枝だけからなり，$E"-T"$の枝の向きがすべてそろった有向ループ:
$T"$はグラフ $G"$の最短木であるから，このループの値は非負である。

補題 8 から次の補題 9 が得られる。

補題 9: グラフ G'から $T"$のすべての枝を任意の順に短絡除去したグラフでは，枝の向きのそろった有向ループの値はすべて非負である。

補題 8, 9 より，グラフ G から始めて 1 回目は枝 E_f, 2 回目以降は最短木の木枝の短絡除去を繰り返すことによって，G の全ノードが 1 点に集約されたグラフ G^+を作ることができる。この過程で短絡除去された枝はグラフ G の木枝 T_f となりグラフ G^+の枝(自己ループ)はグラフ G における T_f の補木枝となる。そして，この変換ではループの長さは保存されるから，グラフ G 上の木 T_fの，基本ループの値はグラフ G^+の各枝の長さと等しく非負である。又，補題 2 より T_fの基本カットセットの値はすべて非負であり，T_fが求める木であることがわかる。

[木の構成アルゴリズム]

step1: グラフ G^*においてsから t へのコスト最小の最大フロー**f**を求め，$T=E_f$ とし，T がグラフ G の木なら停止。

step2: グラフ G の枝 E_fを短絡除去したグラフ G'を作り，r をグラフ G'の任意のノードとする。

step3: グラフ G'において，r から到達可能な全ノードへの最短経路を求め，経路中に含まれる枝の集合を $T"$とする。

step4: $T=T \cup T"$とし，T がグラフ G の木であれば停止

step5: グラフ G'から $T"$の枝を短絡除去し，それを改めてグラフ G'とおき，グラフ G'のノードの内，それから出る枝を持つ任意のノードを r とする。

step6: step3 へ行く。

[例題] 図 3.18 のグラフ G を考える。各枝の数字は長さと重みである。$x(v_i)$, i=1, …5 はそれぞれ 5, 0, −5, 0, 0 であるのでグラフ G^*は図 3.19 となる。ここで数字はコストと容量を表している。グラフ G^*には図 3.20 のsから t へのコスト最小の最大フロー fが存在し，s, t につながる枝を飽和させている。このフローから決まるグラフ G 上の枝の集合が E_fである(図 3.21)。この例では E_f はグラフ G の木にならないので，グラフ G において E_fの枝を(v_2, v_3), (v_1, v_2)の順に短絡除去してグラフ G'をつくる(図 3.22)。このグラフ G 上のノード v_4 から他の全てのノードへの最短経路が図 3.23 のようになり，この経路中の枝(v_5, v_6), (v_4, v_5)を短絡除去するとグラフ G が図 3.24 のようになる。この過程で短絡除去された枝を E_fに追加して作ったグラフ G 上の木が図 3.25 で，図 3.24 の枝(自己ループ)の長さが T の基本ループの値を表している。また，木 T の基本カットセットの値は(v_1, v_2), (v_2, v_3)がそれぞれ 5 で他の枝に対しては 0 であるが，これは図 3.20 の各枝のフローの値と一致している。

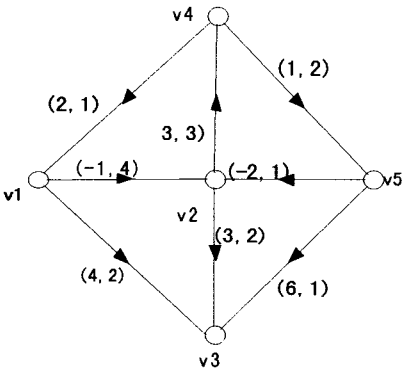


図3.18 グラフG[数字は(長さ,重み)]

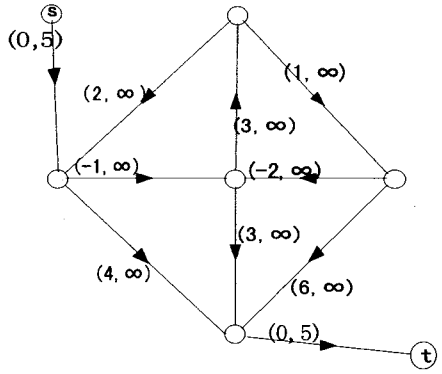


図3.19 グラフG'[数字は(コスト,容量)]

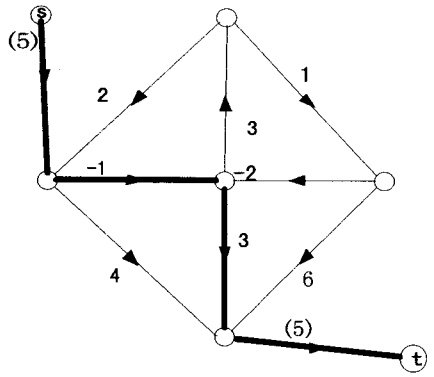


図3.20 コスト最小の最大フロー[(フロー),コスト]

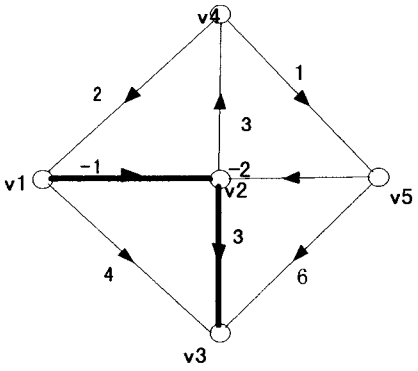


図3.21 B[数字:長さ]

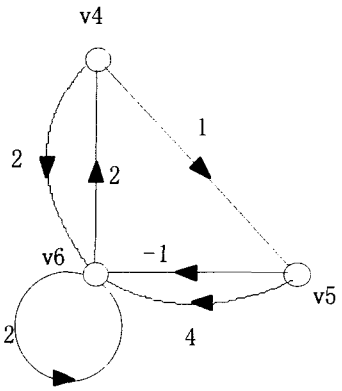


図3.22 グラフG^*

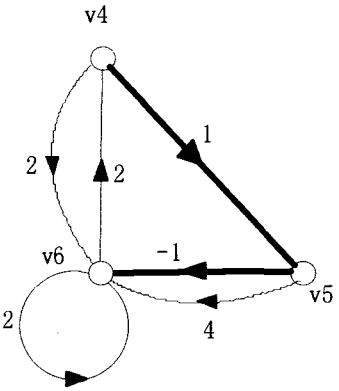


図3.23 最短経路

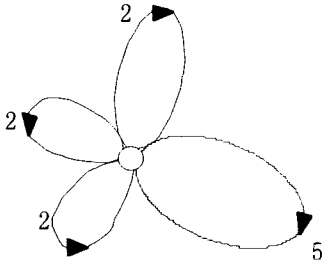


図3.24 グラフG'

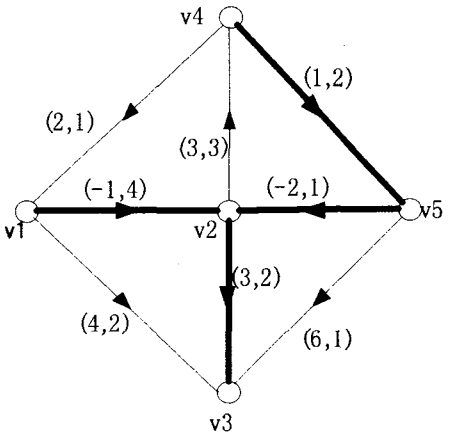


図3.25 木T[数字は(長さ,重み)]

3.6 応 用

本文で提案した手法はコンパクション以外の最適化問題に適用することができる。そのうちのいくつかを次に示す。

ポンプ圧力決定問題

この問題は，水道管網内を流れる水の量が固定され，各管路における圧力の損失(電気回路との対応では電圧降下)が与えられた時，ポンプ，バルブをコントロールして，各ノードの圧力を上下限内に収める問題である その際の目的関数としては，ポンプ圧力の重み付の総和がとられる。

この問題は変数として，ノードのポテンシャル，ポンプ圧力，バルブ圧力，さらに，ノード圧力上下限，ポンプ圧力上限に関するスラック変数を含む線形計画問題となるが，この問題も，上記変数に対応する枝を持つグラフ上で，基本カットセット，基本ループ条件を満たす木を求める問題に帰着され，木の初等変換によって解くことができる。

PERT

PERT では各事象間の関係をアローダイアグラムで表現し，そのうえでクリティカルパスを見つけることによって全工程に要する時間を計算するが，ここでは，次の最適化問題を考える。

各工程 i の所用時間 t_i を一定の範囲内で短縮可能とする。すなわち，t_{i0} を定数，u_i を

$$0 \le u_i \le u_i^*$$

の値を取る変数とした時，工程 i の所用時間 t_i を

$$t_i = t_{i0} - u_i$$

とする。このとき，全工程を指定時間 T 以内に終了し，かつ次式で与えられる u_i の重み付の和

$$\sum_i c_i u_i \rightarrow \text{最小} \quad (c_i \text{は定数})$$

を最小とする問題を考える。ここで，u_i は残業に相当する時間短縮量，c_i はその単位コストと考えることができる。この場合も，一般のアローダイアラムを拡張したグラフ上での基本ループ，基本カットセット条件をみたす木を求める問題となる。

3.7 結言

本章ではネットワーク型問題の例として総配線長最小コンパクション問題を取上げ，線形計画問題のグラフ理論的解釈を行った。そして，コンパクション問題がグラフ上で基本カットセット，基本ループの条件を満たす木を求める問題に帰着されることを示し，木の初等変換および最小コストフローアルゴリズムによって解く方法を提案した。総配線長を最小化する問題は実用的にみても重要であり，このアルゴリズムの効果は大きいといえる。また，ここで示した線形計画問題をグラフ的に解く手法はコンパクション問題だけでなく，他の問題への適用が期待される。

第 4 章 論理合成

本章では論理設計 CAD の分野で重要の問題の一つである論理自動合成を取り上げ，知識処理とアルゴリズムに基づく解法について考察する。この手法は，知識処理とアルゴリズムを組み合せることによって，高速な処理を実現しており，しかも変換規則を入替えることによってプロセステクノロジーの変化にも追随する柔軟性を有している。熟練技術者により設計され，既存の計算機の中で使用されている回路に対して人手設計と比較評価した結果，ほぼ同等の解が数分以内に得られたことを示す。

4.1 緒言

VLSI の多様化，大規模化に伴ない VLSI 設計において CAD は不可欠なものとなっている。特に VLSI を短期間に誤りなく設計するという観点からは論理自動合成の果たす役割は非常に大きい。ただ，一口に論理自動合成といっても動作レベルの記述を記述しアーキテクチャ設計まで扱うものから，低いレベルのレジスタトランスファ記述(以下 RT 記述と呼ぶ)あるいはブール式等を入力し特定のプロセステクノロジにマッピングすることを目的とするものまで各種各様である。当然，まだ研究段階のものが多く，実用のレベルに達しているのはほとんど後者である。本文では，実用化を目的とし後者の問題を扱う。

この問題に対して様々なアプローチが提案されている。これらのアプローチは次の 3 つのグループ: (1)ルールベースに基づくアプローチ[22]，(2)変換テーブルを用いたアプローチ[23, 24]，(3)アルゴリズムを用いたアプローチ[25, 26]に分けることができる。これらのうち，ルールベースに基づくものおよび変換テーブルに基づくものは各種プロセステクノロジーに対するマッピングには適しているが，最適化の能力に関しては部分的な最適化しか行なうことができず，入力された記述に冗長性があった場合でもそれが最適化されないまま出力されることがある。一方，論理最小化アルゴリズムは，2 レベル論理の最適化能力がかなり高く，PLA の設計には向いている。しかし一般の多段回路を設計する場合，その解の良さは使用するテクノロジーにプロセスマッピングするまで評価できず，入力記述が人手により最適化されている場合，かえって悪い結果を導くことがある。従って，これらのアプローチを組み合わせ，それぞれの特徴を生かしたアプローチが望まれる。

本文では，これらの手法を組み合わせたアプローチを用いた論理合成システムについて述べる。すなわち，変換ルールとして(被変換パターンと変換パターンを登録した)変換テーブルを用い，さらに論理最小化アルゴリズムをテクノロジーに依存しない変換ルールとみなして，ルールの解釈実行プログラムにより論理回路の合成を行なう。従って，変換テーブルのみの場合に比べ，論理最小化による最適化機能が強力であり，しかもそれを変換ルールにより物理回路に換し評価できるため，より正確な最適化が可能となる。

4.2 問題の定式化

(1)入力言語

論理自動合成問題の難しさは入力となる記述のレベルに大きく依存する。本文では入力言語として FDL(Functional Description Language)[27]を用いる。これは RT(Register Transfer)記述言語で，VLSI 設計のための機能・論理シミュレーションに用いられている。この言語はシステム内の入出力端子，内部端子，メモリ/レジスタなどの記憶素子，マクロモジュール等の間のデータの転送をその条件と共に記述する。また FDL は記述性を高めるため CASE 文，IF 文を許している。FDL で使用可能なキーワードの代表例を次に示す。

FDL キーワード

- INPUT，OUTPUT，INOUT
- REGISTER，TERMINAL，MEMORY，MODULE
- CASE_OF ，IF_THEN_ELSE
- +，*，XOR，XNOR，
- ADD，SUB，COMBINE，EQ…

次に FDL による記述例を示す。

```
*EXAMPLE
  INPUT  SET, CLK, D, E, X;
  OUTPUT  A;
  REG  A = IF  SET  THEN  1
                    ELSE IF  CLK.UP.  THEN  C
                             ELSE  NOC;
          C = CASE  X  OF
                /0/ D*E
                /1/ D+E;
```

これはレジスタ A とターミナル C へのデータの転送を記述した例である。 DDL や CDL[28]よりも回路の論理的 J 構造を明確に記述しており，論理自動合成向きの言語であるといえる。なお，ここで扱う論理自動合成においては全ての記憶素子は明確に指定されているものとし，その削減などの最適化は考慮しない。

(2)制約条件

制約条件として以下の二つを設定する；

①出力される回路の論理が入力で記述された論理と等価であること，および

②レジスタからレジスタまでの(節点数の意味での)最長経路長が指定値以下であること。

(3)目的関数

目的関数としては出力する論理回路の構成要素の重みの和を最小とすることとする。通常，重みとしては回路のセル数，面積等を選ぶ。

従って，問題は，(2)の制約条件の下で，(3)の目的関数を最小とする回路を求めることとなる。

4.3 論理合成算法

(1)処理概要

まず入力記述(FDL)を中間コードと呼ばれるネットワーク構造に変換する。次に，定数との論理演算，不用信号の除去などテクノロジーに無関係な最適化を行う。これらの処理は，システム組み込みのアルゴリズム型ルールにより実現している。これは，C 言語で記述した関数(サブルーチン)の集合であり，各関数は条件部と処理部から構成することにより処理の高速性と拡張性を実現している。このネットワークをルールの適用及び論理最小化アルゴリズム[29]の実行によって特定のテクノロジーにマッピングする。最後に，レイアウト及びシミュレーション用のデータを出力する。

(2)中間コード

中間コードは論理的な構造を表わすネットワークである。ここで，ノードは論理オペレータ，物理的な回路あるいは端子に対応する。アークはこれらのノードの間のデータの転送を表わす。これらのノードとアークにデータ転送の条件(ビット幅，極性(否定演算の有無)，端子番号，遅延情報)などを割り当てる。

グラフ	属　　性
ノード	演算子，ビット幅，遅延情報
アーク	ビット幅，極性(否占の有無) 始点/終点側端子番号，多重度

このネットワーク上では，極性の伝播によりインバータを削減を容易にするため，否定要素はノードではなく，アークのラベルとして表現する。これにより，中間コード(ネットワーク)の構造をかえること無くインバータの削除が可能となる。また，アークの多重度はその信号の FAN-OUT 数を表す。図 4.1 に，前記 FDL の記述に対応する論理ネットワークを示す。

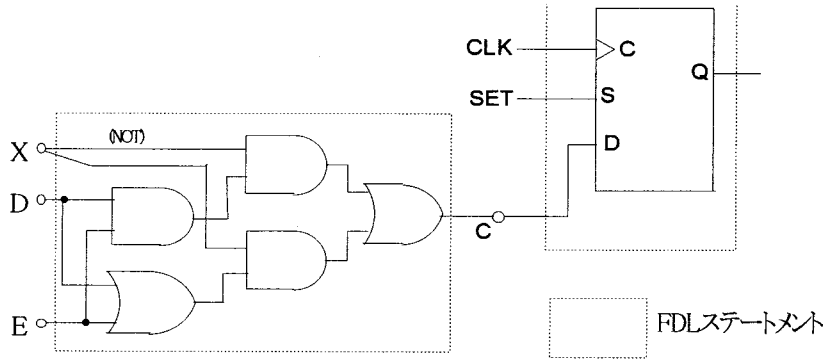
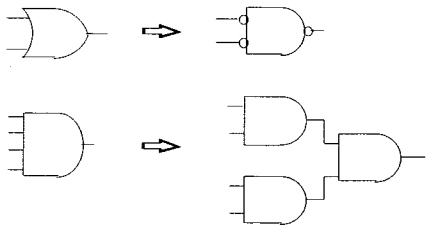


図 4.1 中間コード

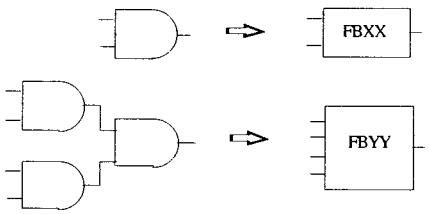
(3)変換ルール

論理ネットワークを物理的な回路に変換するための規則を 2 つの部分ネットワークすなわち被変換パターンと変換パターンで表現する。各ノード，アークは前述の中間コードと同じ属性を持っている。この変換規則には 2 種類のタイプがある。一つは論理ネットワークから論理ネットワークへの変換である。もう一つは論理ネットワークから物理回路(ネットワーク)への変換である。その例を図 4.2 に示す。

変換時には，中間コードと被変換パターンのパターンマッチングを取る必要がある。しかし，任意の変換パターンに対してパターンマッチングを行なうのは困難である。すなわち，この問題はグラフの同定問題を含むだけでなく，不完全な対応関係をも発見しなければならない。例えば，中間コード上で 2 入力の AND ゲートと非変換パターン内の 3 入力の AND ゲートは対応づける必要がある(3 入力の AND ゲートの場合入力線の 1 本を常に 1 とすると 2 入力の AND ゲートの機能を果すことができる)。そのため，ここでは非変換パターンを木に限定する。また論理最小化アルゴリズムは論理ネットワークから論理ネットワークへの変換を表わす特殊なルールであると見なして処理を行なう。



タイプ1: オペレータ→オペレータ



タイプ2: オペレータ→物理回路

図4.2 変換規則の例

(4)マッピング: 物理回路への変換

このシステムは変換テーブルと論理最小化アルゴリズムを組み合わせて用いる。すなわち，図4.3の探索木の各ノード(ルールの適用を意味する)を調べることにより求める解を発見する。

① 探索木と部分回路の切り出し

本手法では計算時間の増大を抑えるため変換規則の変換パターンを木に限定する。このとき，中間コードにおいてfan outが2以上のノードは変換パターンの出力ノードとのみ対応づけられることになる。そのため，変換にあたっては，中間コード(ネットワーク)をfan outが2以上のノードの出力アークを除去することによって分割されるいくつかの部分ネットワーク毎に変換を行なえば良い。

実際のFDL記述では，各ステートメントの左辺に現れる中間変数は他のステートメント内で2回以上使用されることが多い。すなわち，その中間変数に対応するノードのfan outは2以上となる。そこで，ネットワーク全体からFDLの1ステートメントに対応する部分回路を切り出し，それを順に処理することにより計算時間の短縮をはかる。前記FDL記述の例では2つのステートメントが記述されており，これらが一つずつ処理される。なお，この場合，全回路にわたる最適性が問題となるが，人手設計ではほとんどの場合FDL 1ステートメント毎に行なわれており，いくつかの例題で検証した結果でも問題はないことが確認された。

② 解の探索範囲の限定

FDLの1ステートメントに対応する部分回路の変換について述べる。論理最小化アルゴリズムは論理的な意味で最小化は行うものの，物理的な回路への変換を考慮していないため，実際に変換テーブルを用いて最終的な回路に変換するまで本当に良いかどうか判断できない。そのため，このアプローチでは論理最小化アルゴリズムを1種のルールとみなして処理する(図4.3探索木参照)。

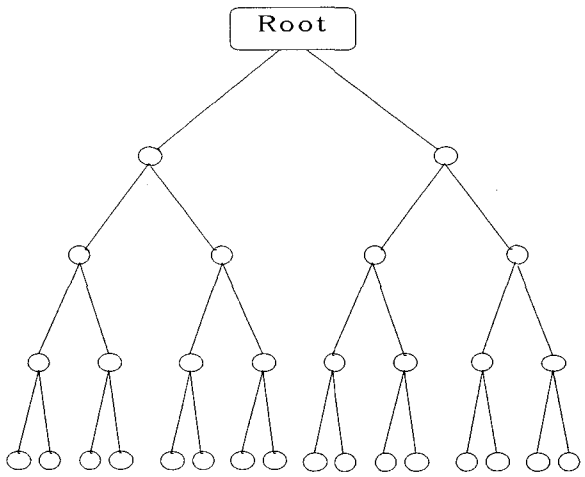


図4.3 探索木

この変換はルール解釈実行プログラムが管理する。このプログラムはまず，未処理のノードのうち出力側の隣接ノードがすべて変換済みとなっているノードの一つを次に変換するルートノードとして選択する。次に，ライブラリ中の変換ルールを調べ，ネットワークのトポロジー，FAN-IN/OUT，極性，遅延時間等の条件を満足するものを逐次適用する。このとき無駄な探索を排除するため，すべてのルールをいくつかのグループに分け，各グループの中ではルールごとに定義したコスト(実際はゲート数を用いる)に基づいてソートしている。従って，例えば2入力のANDの論理を実現する場合，2入力のANDゲートで変換可能な場合は(同種の論理でコストの大きい)3入力のANDゲートがあったとしても適用を除外する。なお，前記の通り否定の演算はアークのラベルとして表現しており，インバータ削減のため極性の伝幡を考慮している。また，各素子のFAN-OUTは常にチェックしており，必要に応じてバッファを挿入している。

図4.4に解の探索の例を示す。ここでは3個の可能な論理回路が生成されている。ルール解釈実行プログラムはこれらをすべて調べコスト最小の回路(通常はゲート数最小の回路)を解として選択する。

③ 遅延時間の考慮

変換の際，ネッワーク上でレジスタ，入/出力端子間の最長経路を計算することにより遅延時間の考慮を行う。

まず，中間コードの各ノードに遅延時間の予測値を割り当て，入力端子またはレジスタの出力端子から各ノードへの最長経路長を計算する。そして，各ステートメントは1個ずつ順に処理することは前記のとおりであるが，その処理順を出力端子側からとし，各ステートメントが処理される時点で，そのステートメントに対応する信号が到達する全てのノードは既に変換済みであるとする。従って，出力端子またはレジスタの入力側端子への最長経路長はライブラリの特性値に基づいて計算できる。そのため，あるノードが処理された場合，そのノードから出力方向への最長経路の値と，入力側からそのノードへの最長経路の予測値を加え，それが指定値を越えた場合，その変換をスキップする。出力側への最長経路長は変換の過程で容易に更新することが可能である。

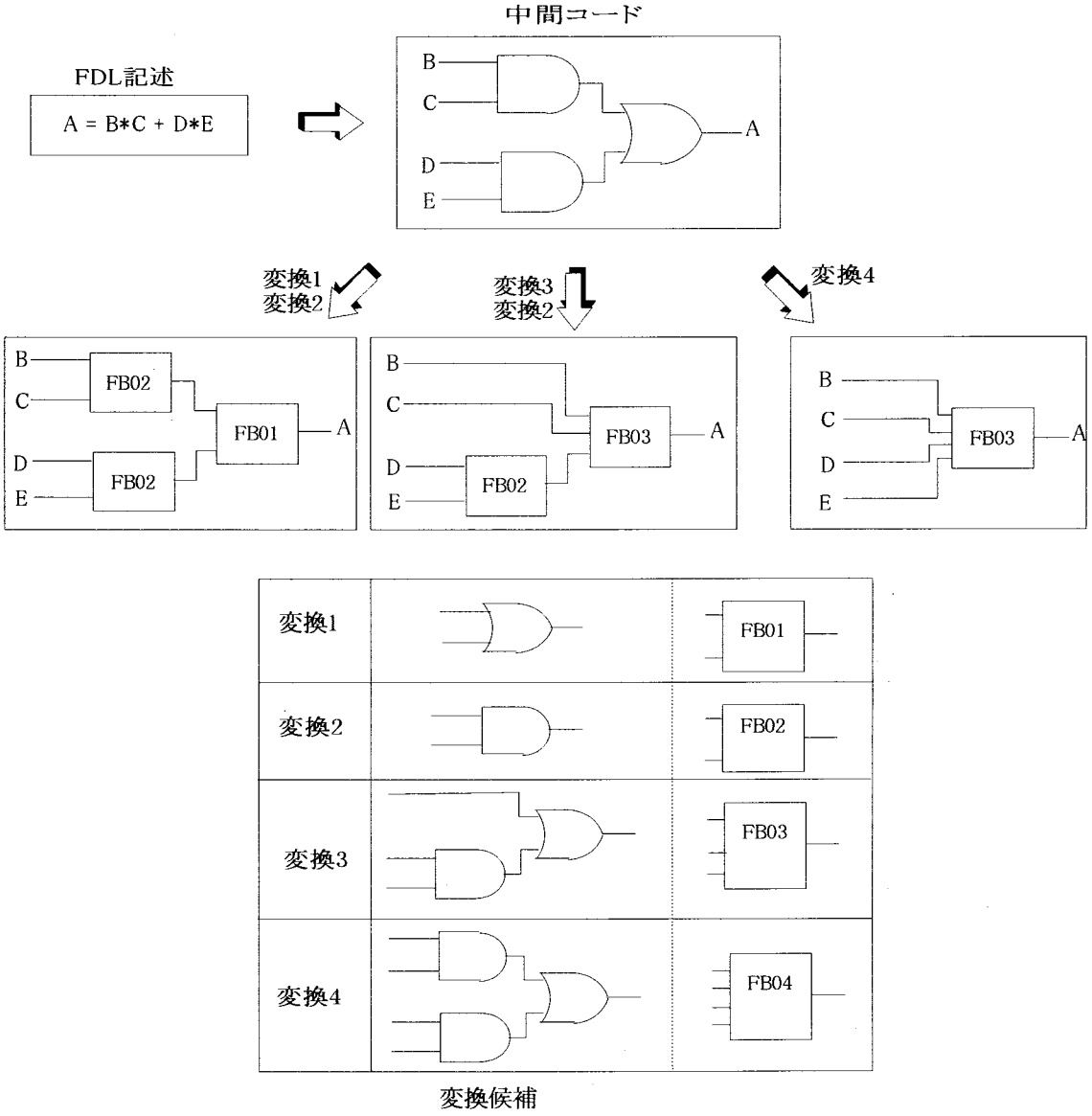


図 4.4 適用可能な変換の例

4.4 算法の高速化

① 探索打ち切り条件

ルール解釈実行プログラムはすべての生成可能解を調べるので各 FDL1 ステートメントの処理を行う際，探索打ち切り法が不可欠である。本文では以下に述べる効率的な探索打ち切り手法を提案する。

まず，探索木のトレースを DFS(Depth First Search)によって行うものとする。このとき，探索の途中のネットワークは図 4.5 に示すとおり，出力側の既変換部と入力側の未変換部に分けられる。

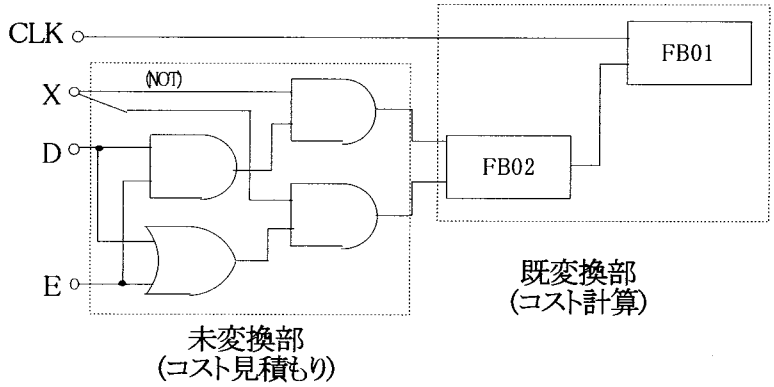


図4.5 コストの予測

問題は未変換部の変換に要するコストの下限値を求めることである。そのため，FDL の 1 ステートメントに対応するネットワーク上で任意の木を仮定する(図 4.6。この木は，FDL の特性から，このネットワーク上で最も出力側にあるノードをルートとする ROOTED TREE となる。この木によって各ノードにひとつのサブネットワークを対応させることができる。

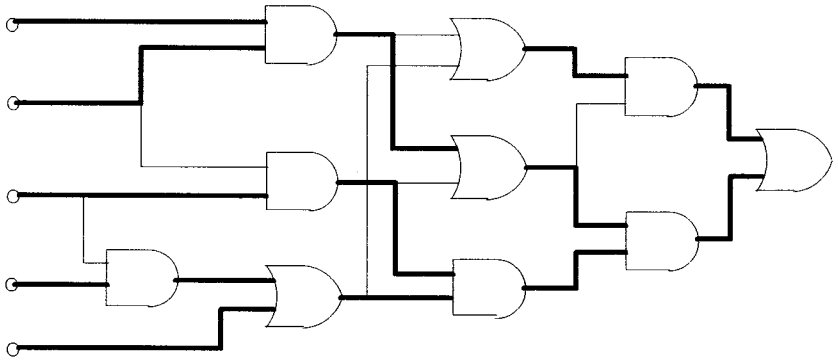


図 4.6 木とサブネットワーク

そこで，ノード n に対応するサブネットワークを $G(n)$とし，そのサブネットワーク $G(n)$を物理回路に変換した時のコスト(ゲート数)の下限を $LB(n)$とする。

$LB(n)$の値は

$$LB(n) = \min_{i \in R}(C(i) + \sum_{j \in N(n,i)} LB(j))$$

で計算できる。ここで，R はノード n に適用可能な全てのルールの集合，$C(i)$，$N(n, i)$はそれぞれノード n をルール i で変換した時のコストおよび，その結果生成される変換パターンの入力側に隣接する $G(n)$のノード集合である。LB の値は探索木のトレース順が DFS であることから変換の途中で容易に決定することができる。

つぎに，変換の途中でネットワーク中のすべてのノードを図 4.7 に示すように 3 個の集合に分けて管理する。すなわち，グループ 1 の集合はすでに物理的な回路に変換済みのもの，グループ 2 のノードはグループ 1 に隣接しているノードの集合，そしてグループ 3 はその他のノード集合である。変換に際してグループ 2 及びグループ 3 のノードに対する *LB* の値を上記の式によって更新する。このとき，グループ 2 のノード集合の *LB* の値の和は未変換のノード，すなわちグループ 2 及びグループ 3 のノードを変換するのに要するコストの下限値を表わしている。そこで，ルールの適用に際して変換の途中で常にこの値を計算す k ことにより解の探索を続けるか否かを決定する。 計算機実験によると 90%以上の探索が削除されることがわかった。

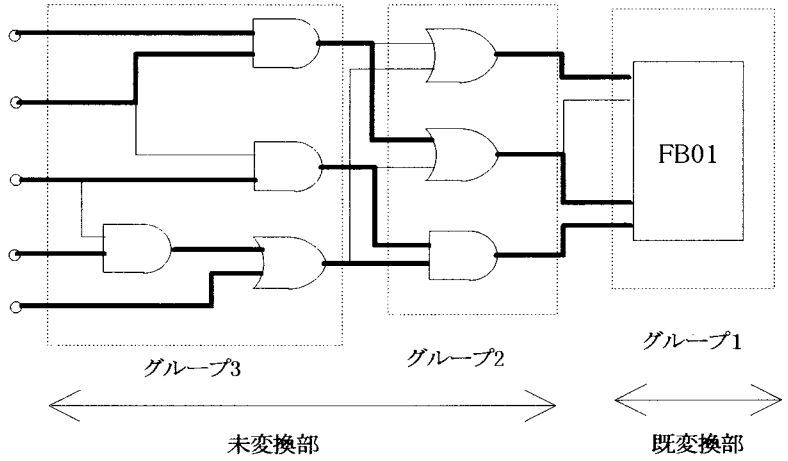


図 4.7 ノード集合の分類

4.5 考察

このシステムを C 言語で作成し，計算機実験を行なった。実験用データとしては人手で設し実際に使用している回路を用い，変換ライブラリとしてバイポーラと CMOS 回路を用いた。

(1)実験 1

システムに組み込まれた論理最小化アルゴリズムの効果を見るために実験を行った。この実験によると論理最小化を組み込むことにより約 20%の解の改善が行われた。論理最小化アルゴリズムは特に CASE 文，IF 文等の条件文の処理に効果があった。これは，条件文を and，or により展開するとき回路が冗長となりやすいことによるものと思われる。

(2)実験 2

実際の計算機に使用されている回路をテストデータとして用いた。表 4.1 にこのシステムで生成された結果と人手設計結果の比較を示す。

表 4.1 人手設計との比較

データ	自動設計	人手設計	比率	時間
EX.1	400 セル	406 セル	99%	94.6 秒
EX.2	398	474	84%	85.5
EX.3	562	513	110%	42.2
EX.4	128	133	96%	9.6
EX.5	975	1272	77%	132.0
EX.6	3569	3455	103%	315.0
EX.7	119	128	93%	30.7
EX.8	175	148	118%	27.2
EX.9	1530	1408	109%	180.0
EX.10	1114	1408	79%	165.0

この結果は本システムが人手設計と同等規模の回路を短時間で生成していることを表している。なお，既に述べた通り上記の結果はFDL ステートメントを 1 個ずつ順に処理して得たものであるがいくつかの例題に対して全ステートメントをまとめて処理してみた。しかし結果はほとんど同じであった。その理由は入力されたFDL レベルで中間変数の抽出がうまく行なわれていることによるものと思われる。

4.6 結言

本章ではルールベースと論理最小化アルゴリズムを組み合わせたアプローチに基づく論理最小化システムについて述べた。特に，効率的な解の探索打ち切り条件により，両手法の組み合わせを可能とし，短時間に人手設計と同等の回路を生成可能とした。このシステムはすでに実用化され，実際の回路設計に使用されている。

第5章　管路網解析

本章では水道網解析問題を考察する。この問題は水道管網の取水量や需要量，ポンプの圧力等の条件が与えられた時，管網内の流量および圧力を計算するものであり，流量および圧力を電流および電圧，ポンプを電圧源，取水点および需要点を電流源，管路を非線形抵抗にそれぞれ対応させると非線形回路網解析問題となる。この問題の解法としてニュートン法が有力であるが，初期解の与え方によっては解の収束性に問題が生ずるという欠点があった。そこで，本文では各管路の非線形特性を折れ線近似して管網解析を行なう手法を提案する。この手法は折れ線近似による電気回路解析手法であるKatzenelson法を次の点で改良し，処理時間の短縮を実現したものである。(1)まず，粗い近似からはじめてその精度を徐々に更新していき，最終的に必要な精度の解を得ること，(2)各管路の非線形特性に注目し，折れ線近似の区間幅および区分点位置を管路の初期流量に基づいて決めることにより反復回数を大幅に減らしていること，(3)行列演算の高速化のためLU分解をデータ領域の初期化と数値計算の二つの部分に分け，数値計算部において行列要素の計算順序を工夫することによりポインタ操作の手間を減らしていること。

また，疎行列に対する新しいLU分解手法をあわせて提案すループログラム実験の結果，提案した手法は現実の水道網を含む全ての例題について収束し，計算時間も従来の手法に比べ約1/20倍であったことを示す。

5.1　緒言

上水道における配水管網は配水池，取水点分岐点需要点とそれらの各施設を結ぶ管路からなっている。管路に取込まれた水は一旦配水池に貯水されたあと管路と通じて各需要点へ送られる。その制御は管網内に設けられたポンプ，バルブによって行なわれる。

管網解析問題は管網の特性，ポンプ，バルブの圧力，取水量，需要量が与えられた時，定常状態での管網内の流量圧力を求める問題である。最近，水道事業の分野でも省資源，省エネルギーが重要な問題となり，大規模な管網の解析を高速に実行したいという要求が高まってきている。

管網解析手法としてはこれまでNewton法に基づく方法[30]や数理計画法を利用した方法[31]などが提案されている。このうちNewton法は最も広く用いられる方法で，良い初期解を与えると収束は非常に速い。反面，初期解の与え方によっては収束しないという欠点がある。これに対して，数理計画法を利用した方法では，各管路の非線形特性を階段関数で近似し，最小コストフローアルゴリズムを用いて解く方法で，行列を扱う必要がなく，収束も保証されることなどが特徴である。しかし，階段関数による近似を用いているため，実用上十分な精度をもつ解を得るには区分点の数を多く取る必要があり，計算時間の点では不利である。

本文では電気回路解析手法の一つであるKatzenelson法[32]を改良した手法を提案する。Katzenelsonh法は非線形関数を折れ線近似して解く方法で，収束が保証されるという大きな利点があるが，近似精度を上げるため区分点数を増やすと計算時間が増大し，そのままでは大規模な管網の解析に使うことはできない。そこで本文で提案する手法は，Katzenelson法に次の点で改良

を加え計算時間の大幅短縮をはかったものである。(1)折れ線近似の区分点数を徐々に増やす。すなわち，最初は粗い近似で解き，その結果を初期解としてより精度を上げた近似のもとで解く。これを繰り返して最終的に必要な精度をもつ解を得る。(2)各管路の非線形特性を表わす関数型に注目し，折れ線近似の区分点位置を各管路の初期流量に基づいて決定する。(3)扱う行列の構造が計算の過程では不変であることを利用して行列演算の高速化をはかる。

また，本手法は一部の変更によってNewton法にもなる。従って，まず折れ線近似のもとで近似解を求め，それを初期解としてNewton法を適用することにより，厳密解を高速にしかも，ほぼ確実に求めることも可能である。

以下，まず水道管網を電気回路網に対応づけて問題の定式化を行ない，次に管路の非線形特性を折れ線近似した高速解法について述べ，最後に計算機を用いた実験により本手法の有効性を示す。

5.2 問題の定式化

(1)管網

管網は取水点，分岐点，配水池，需要点とそれらの各施設を結ぶ管路からなっている。取水点は外部から水を取込む点，配水池は水を貯めるおくための池，分岐点は管路と管路の接続点，需要点は水の消費点で水はここから外部に流れ出していく。管路内の特定の管路には水の流れを制御するための，ポンプ，バルブが設定されている(図5.1)。

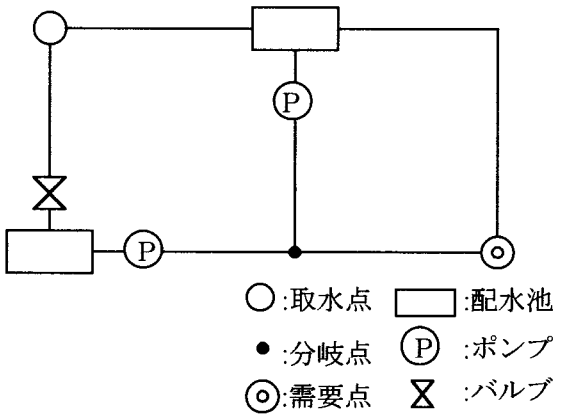


図5.1 水道管網の例

管路は円筒形をしており，その流量と損失圧力の関係は次式で表わされる[33]。

$$q = 0.2783 c d^{2.63} \ell^{-0.54} h^{0.54} \qquad \text{(Hazen-Williams の式)}$$

ただし，

q: 流量(m^3/sec) h: 損失圧力(m)

d: 管径(m)　　ℓ: 管長(m)

であり，c は管路の材質や径年変化などによって決まる定数で通常 100 前後の値である。

(2)電気回路によるモデル化

水の流量を電流，圧力を電圧に対応づけ，管網の各要素を次の電気回路素子でモデル化する。

管路　: 非線形抵抗(電圧電流特性は管路の圧力，流量特性に等しい)

ポンプ: 可変電圧源(起電力はポンプ圧力)

バルブ: 可変電圧源(起電力はバルブによる損失圧で，向きはポンプと逆)

需要点: 接地した電流源(電流値は需要量)

取水点: 接地した電流源(電流値は取水量)

配水池: 接地した電圧源(起電力は配水池水位)

例えば図 5.1 の回路は図 5.2 の回路でモデル化される。問題はこのようなモデル化した電気回路上で KCL(Kirchhoff's Current Law)，KVL(Kirchhoff's Voltage Law)を満たす電流，電圧を求めることである。

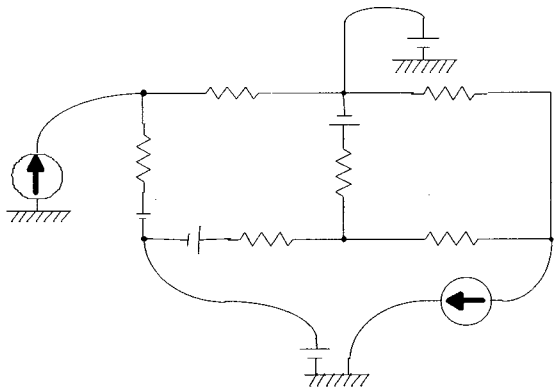


図 5.2 管路網の電気回路モデル

(3)回路方程式

回路方程式の立て方にはループ解析法，カットセット解析法，節点解析法があるが，ここではアルゴリズムの容易さを考え節点解析法を用いる。そのため回路の各節点と ground の間には抵抗または電圧源の枝からなるパスが少なくとも 1 本存在するものとする。(この条件は管網の各連結成分ごとに配水池が 1 個以上存在すれば満たされるが，そうでない連結成分があればその中の適当なノードを 1 個接地すれば良い。)回路中の配水池に対応するノードの集合を V_0，それ以外の施設に対応するノードを V_1，管路に対応する枝の集合を E とすれば回路方程式は次の様になる。

$$i_{ab} = g_{ab}(\psi_a - \psi_b + e_{ab}); \quad (a,b) \in E \tag{5.1}$$

$$\sum_b i_{ab} = j_a; \quad a \in V_1 \tag{5.2}$$

ここで i_{ab} は枝(a, b)の電流値，ϕ_a はノード a の電位である。ただし V_0 の各ノードの電位は定数である。また g_{ab} は非線形抵抗枝の電圧−電流特性を表す関数で sgn(v) を v の符号，k_{ab} を枝(a,b)固有の定数とする時

$$g_{av}(v) = k_{ab}\,\mathrm{sgn}(v)|v|^{0.54} \tag{5.3}$$

で与えられる。e_{ab} は枝(a, b)に電圧源がある時にはその起電力とし，ない時には 0 とする。j_a はノード a に電流源がつながっている時にはその電流値とし，そうでない時には 0 とする。なお，i_{ab}，e_{ab} の符号はその向きがノード a から b の時を正，逆の時を負とする。また j_a の符号はノード a に入る向きを正とする。

独立変数を各ノードの電位とし，(5.1)式を(5.2)式に代入すると

$$\sum_b g_{ab}(\psi_a - \psi_b + e_{ab}) = j_a; \quad a \in V_1$$

となる。変数の数は V_1 に含まれるノード数であり，式の数と等しいので，問題は非線形連立方程式を解くこととなる。

5.3 区分線形近似による管網解析手法

(1)基本的な考え方

解くべき式は非線形連立方程式であるが，現れる非線形関数は(5.3)式だけである。枝 j における電圧の適当な初期値を $v_j(0)$とし，$v_j(0)$を含む区間$[l_j(0), u_j(0)]$を用いてその枝の非線形特性を直線近似する(図 5.3(a))。全ての枝の特性を同様に近似すると，解は連立一次方程式を解くことにより容易に求めることができる。その時枝 j にかかる電圧を $v_j(1)$とする。$v_j(1)$は必ずしも区間 $[l_j(0), u_j(0)]$内にあるとは限らないが，いま，全ての枝について，区間内に入っていると仮定する。そこで，枝 j の近似を $v_j(1)$を含む区間 $[l_j(1), u_j(1)]$を用いて図 5.3(b)のように更新する。ただし

$$u_j(1) - l_j(1) < u_j(0) - l_j(0)$$

である。

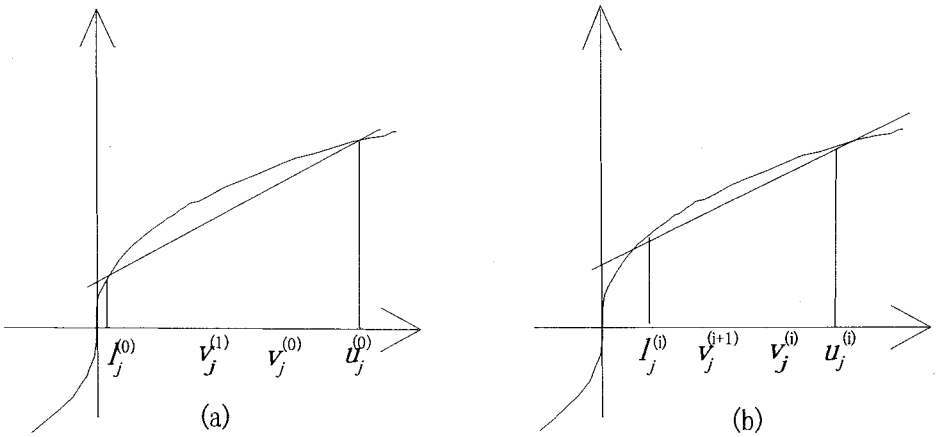


図 5.3 非線型関数の近似

同様に他の全ての枝の近似を行なって連立一次方程式を解く。そして，各枝 j にかかる電圧 $v_j(2)$ が再び区間$[l_j(1), u_j(1)]$にあったと仮定する。次に，幅のより小さな区間$[l_j(2), u_j(2)]$を定義して同様の処理を繰り返す。

さて，各枝の電圧 $v_j(i+1)$が常に区間 $[l_j(i), u_j(i)]$に含まれるという仮定のもとでは以上の解き方が考えられる。この場合，非線形関数の近似誤差は区間幅のほぼ 2 乗に比例して小さくなる(後述)ので，区間幅を毎回 1/2 程度に縮小できれば収束は十分速いと言える。もちろん $v_j(i+1)$が常に区間 $[l_j(i), u_j(i)]$に含まれるとは限らないので，実際には各枝の非線形特性を図 5.4 の様に折れ線近似することにより，各枝の電圧を含む区間を常に定義し，Katzenelson 法を適用する。すなわち，枝電圧が近似区間をまたがって変化するごとに連立 1 次方程式を解く。しかし，テストデータによる実験では区間の選択に注意することにより区間幅を 1/2 程度に縮小する場合でも $v_j(i+1)$が区間 $[l_j(i), u_j(i)]$の外にでる確率を 1%以下におさえることができた。

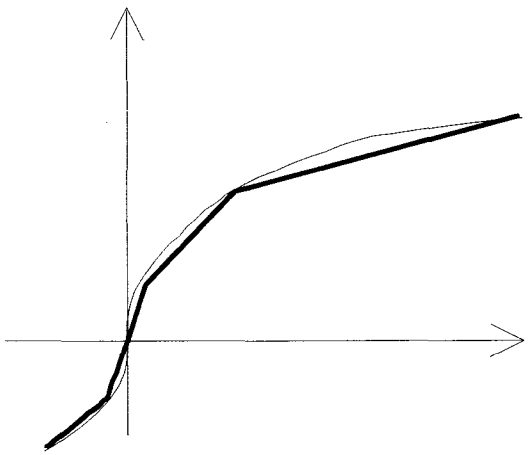


図 5.4 非線形関数の折れ線近似

(2)アルゴリズム概要

アルゴリズムの概要を次に示す。

step1: 各ノードの電位 ϕ の初期値を ϕ_0 とする

step2: 非線形関数の折れ線近似を行なう。

step3: 解くべき式を形式的に

$$f(\phi) = j$$

と書くとき，各ノード電位の初期値 ϕ_0 より残差

$$d = j - f(\phi_0)$$

を計算する。

step4: A を ϕ_0 の属する領域における $f(\phi)$の Jacobi 行列とする時，A^{-1} を求める (LU 分解する)

step5: $r = 1$， $\phi = \phi_0$

step6: $\triangle \phi = A^{-1} d$

step7: $\phi_s = \phi + s\triangle\phi$

とし，sの値を0から増加させて各ノードの電位を変化させた時，各枝にかかる電圧が最初に折れ線近似の区分点に達する時のsの値をs^*とし，電圧が区分点に達した枝をjとする。

step8: $s^* < r$ のとき

(1) $\phi = \phi + s^*\triangle\phi$

(2) $r = r - s^*$

(3) 枝jの近似区間を更新

(4) A^{-1}を更新

を行ないstep6にもどる。$s^* \geqq r$ならstep9へ

step9: $\phi = \phi + r\triangle\phi$

step10: ϕの誤差が指定された値以下なら停止，そうでなければ区分点数を増加して折れ線近似を行ない，$\phi_0 = \phi$としてstep3にもどる。

このアルゴリズムのうちstep8が，各枝にかかる電圧が近似区間に入らない時の処理で，Katzenelson法に基づいている。なお，step2，step10の折れ線近似をϕ_0における接線近似とし，step7，step8を取り除けばNewton法となる。従って，本手法で精度の良い近似解を求め，それを初期解としてNewton法により厳密解を求めループログラムの作成も容易である。

上記アルゴリズムの高速化には次の配慮が必要である。

(a) 各枝の電圧が折れ線近似の区分点をまたがって移動する毎にstep6～step8が実行される。このアルゴリズムで時間を要するのは行列演算であるが，予備実験の結果，現実の管網を扱う場合step6の1回の代入計算およびそれにともなうstep8の枝電圧更新処理にstep4のLU分解の約20%の時間を要することがわかった。従ってstep8の繰り返し回数を減らすことが重要となる。

(b) 各ステップの処理を高速化すること

そのため，以下(a)に関しては折れ線近似区間の設定について，(b)に関しては行列のLU分解の効率化について提案する。

(3)折れ線近似

本文で提案するアルゴリズムは最初は区分点の少ない粗い近似のもとで解を求め，それを初期解としてより区分点の多い近似のもとで解く処理を繰り返している。ところで，前述の通り，3.2のアルゴリズムのstep6～step8の実行回数は各枝にかかる電圧が区分点をまたがって変化した回数に等しい。そこで，1段階前の解を初期解とし，より精度の高い解を求める際，各枝の電圧がその初期値と同じ区間に入るよう折れ線近似の区分点を決めるのが望ましい。そのためには近似の区間幅および区分点の選択に注意する必要がある。

① 区間幅の決定

区間幅は変数の各値に対して誤差ができるだけ一定となるよう決めるのが望ましい。そこで，まず折れ線近似の誤差を見積もることから始める。

各枝の非線形特性は(5.3)式で表されるが，ここでは次式の誤差の評価を行なう。

$$i = \mathrm{sgn}(v)\,|v|^{0.5}$$

i_0(≧0)と d(>0)が与えられた時，区間[v_0，v_1]内の近似誤差を見積もる。ただし，

$$v_0 = i_0{}^2$$

$$v_1 = (i_0 + d)^2$$

である。この区間内では誤差(図 5.5 における $i - i_2$)は

$$e(i) = i - (i_0 + (i^2 - i_0{}^2)/(2i_0 + d))$$

となる。また，この区間内での誤差の最大値は

$$e(i + d/2) = d^2/4(2i_0 + d) \tag{5.4}$$

で与えられる。逆に，誤差を定められた値 $\bar{e}$ 以下におさえるには次式を満たす d の値を選べば良い。

$$d \le 2(e + \sqrt{-\bar{e}^2 + 2\bar{e}\cdot i_0}) \tag{5.5}$$

(5.4)式は折れ線近似の誤差は d の値が i_0 に比べて十分小さい時は d のほぼ 2 乗に比例することを示し，(5.5)式は誤差を一定におさえるときは i_0 が大きくなるほど区間幅を広げて良いことを示している。この結果は正確な非線形抵抗特性を扱って得たものではないが，実用上は(5.5)式に従って区間幅を決めて良い。また，ここでは i_0 が非負の場合，すなわち，原点を含まない場合の評価を行なったが，区間が原点を含む場合の誤差は(5.4)式より小さくなる。

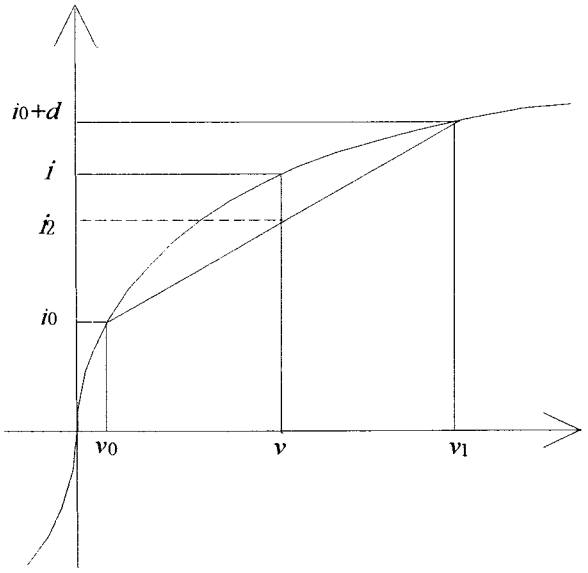


図 5.5 折れ線近似誤差の評価

② 区分点位置の決定

区間を徐々に縮小し近似精度を上げた場合，区間幅を大幅に縮小しなければ，各枝の電圧はほ

とんどの場合，1段階前の解の近傍にあることが予想される。従って，各枝の電圧の初期値 $v(k)$ が与えられた時，その枝の近似区間 $[l(k), u(k)]$ (図 5.6)は $v(k)$の近傍を含む様に決める。

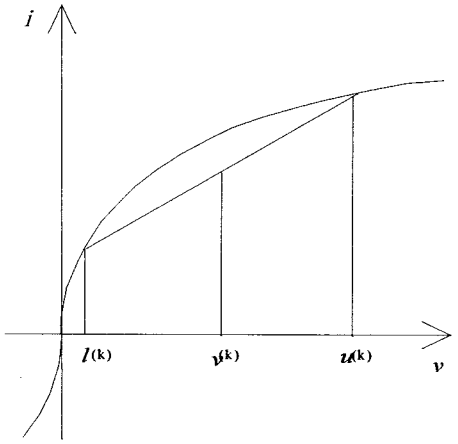


図 5.6 電流の近似値と対応する電圧

③ 関数表を用いた区間幅，区分点位置の決定

以上の考察の結果，本文では次の関数表を使用した区間幅，区分点位置の決定法を提案する。まず，関数近似誤差に関する定数 e をあたえ，(5.5)式に基づいて，次の手順で i_j を決定する。

$$i_0 = 0$$

$$i_{j+1} = i_j + 2(\bar{e} + \sqrt{\bar{e}^2 + 2\bar{e} \cdot i_j}) \ ; \quad j = 0,1,\cdots$$

ついで，次式で各 i_j に対応する電圧 v_j を計算する。

$$v_j = i_j^{1/0.54}; \quad j = 1,2,\cdots$$

これらの値を用いて関数表

$$(i_j, v_j); \quad j = 0, \pm 1, \pm 2, \cdots$$

を作成する。ただし

$$v_0 = 0$$

$$v_{-j} = -v_j; \quad j = 1,2,\cdots$$

$$i_{-j} = -i_j; \quad j = 1,2,\cdots$$

である。アルゴリズムではこの関数表を用いて以下の様に区間幅，区分点位置を決定する。 区間幅，区分点位置を決定するパラメータをそれぞれ w，r(いずれも正の整数)とする。電圧の初期値 v_S に対し j を

$$v_j \le v_s \le v_{j+1}$$

を満たす整数とする時 v_s を含む近似区間は

$$\left[v_{j-aw}, v_{j+bw}\right]$$

とする。ここで a, b は

$$a = rw,\quad b = (z-r)w \qquad v_s > 0 \text{の時}$$
$$a = (z-r)w,\quad b = rw \qquad v_s < 0 \text{の時}$$
$$a = zw/2,\quad b = zw/2 \qquad v_s = 0 \text{の時}$$

である。(z は関数表固有の定数)

(例) v_s>0, j=100, w=2, r=14, z=16 の時の区間は[v_{72}, v_{104}]であり，r=10 の時の区間は[v_{80}, v_{112}]となる。

5.4 疎行列演算手法

前記のアルゴリズムでは近似精度を上げる毎に LU 分解を繰り返す必要がある。従ってこの行列演算を効率良く行なうことが重要となる。ところで，この行列演算で扱う行列はスパースであり，しかも行列の非零要素の位置は管網の構造によって決まるのでアルゴリズムの実行中は不変である。この場合，計算時間の短縮に用いられる手法にコード生成法，アドレス表生成法がある[34]。これらの手法を用いれば行列演算の時間は大幅に短縮されるが，大規模な問題を扱かう場合，発生されループログラムやアドレス表が膨大となり不都合である。そのため本文ではそれにかわる手法として行列の構造(非零要素の位置)が与えられた時点で LU 分解の結果を格納するためのデータ領域の設定(LU 分解の途中で発生する非零要素の領域の確保，ピボット順の決定等)をあらかじめ行ない，LU 分解の繰り返し時の負担を軽くする方法を用いる。以下その手法について述べる。

(1) データ領域の設定

本文で扱う行列は対称であるから行列の上半分のみ処理する，データ領域の設定の部分では Gauss 法を用いて LU 分解をシミュレートすることによりその過程で発生する非零要素の位置を決定しデータ構造を作成する。ここでは数値計算は行なわない。またピボット順の決定には Markowitz 法[35]を用いる。

(2) LU 分解

LU 分解の手法には様々な手法が提案されている。いずれの手法を採用しても行列の対称性を考慮したものであれば数値演算の回数は n×n 行列の場合 $n^3/6$ であるが，本文で用いるデータ領域の設定をあらかじめ行なう方式の場合，ポインタの操作に要する時間にかなりの差が生じる。その理由を Gauss 法を例に取って説明する。

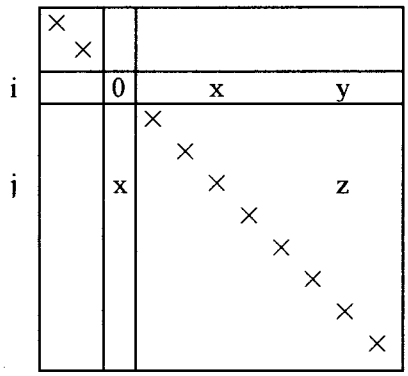


図 5.7 Gauss 消去法

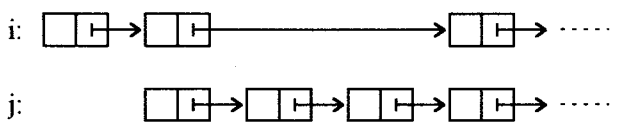


図 5.8 リスト構造の例

図 5.7 で対角要素 d をピボットする処理を考える。Gauss 法では x の値が非零のとき，i 行の各々の非零要素 y に対して，y と同じ列にある j 行の要素 z の更新を行なう。この処理を行なう場合，通常図 5.8 のような 2 組のリストを用いるが，データ領域の設定をあらかじめ行なってある場合，d 以後のピボット処理で発生する非零要素も j 行のリストに登録されているので d のピボット処理では値の変化しない要素が多い。ところが，j 行のリスト上で値を更新する要素の位置を見つけるには値を更新する必要のない要素を経由しなければならず効率が悪い。そこで本文では次に示す処理手順を提案する。

```
for i=1 to n do
    B(i)=0
end
for i=1 to n do
    for j=1 to i−1 do
        for k=1 to n  do
            B(k)=B(k)+u(j, i)*U(j, k)/U(i, i)
        end
    end
    U(i, i)=1/(U(i, i)−B(j))
    for k=i+1 to n do
        U(i, k)=(U(i, k)−B(k))*U(i, i)
        B(k)=0
    end
end
```

ここで扱う行列は対称であるから，初期状態では配列 U には LU 分解される行列の右上の部分および対角項が入っているものとし，最終的には LU 分解した結果のうち上三角行列および下三

角行列の対角項を格納するものとしている。この手法の特徴はUの更新を行なう際，各要素間の計算順序を工夫することによって，更新する値をUに直接書込まず，一旦Bという1次元配列に書込み，一つの対角項をピボットする処理が完了した後，改めてUの値を更新していることである。スパース処理を行なわない時はこの処理は無意味であるが，スパース処理を行なう時，特にデータ領域の設定をあらかじめ行なう場合は効果的である。すなわち，スパース処理を行なう時はUはリスト構造で表現されるため要素の位置を見つけるにはリスト上を探索しなければならないが，Bは1次元配列であるためその必要はない。従って，前記のようにzの位置を探索する処理は不用で配列Bに直接書込める。また配列Uの値を最終的に更新する時も配列Bから直接読み出して行なうことができる。これによりLU分解時のポインタ操作の手間を大幅に短縮することができる。図5.9に本手法における配列の添え字i，j，kの変化状況を示す。

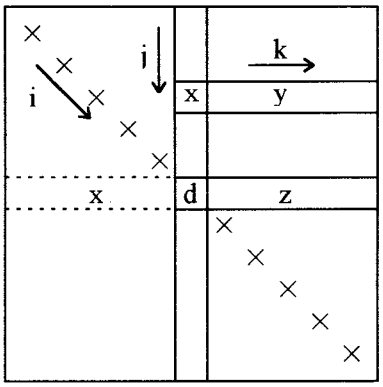


図5.9　LU分解法の説明

5.5　実験結果と考察

本文で提案したアルゴリズムの評価を行なうため，Fortranでプログラムを作成し，計算機実験を行なった。このアルゴリズムは前述の通り部分的な修正でNewton法にもなるので，プログラムは本文で提案した方法，Newton法および両者の組合せで解く機能を有していループログラムの実行に必要な記憶装置の量はノード数500の管網の場合プログラムとデータ領域をあわせて約240kbyteである。なお，使用計算機はACOS900である。

(1)実験データ

現実の管網に相当する規模をもつ4種類のテストデータを用いた。

① データ1

格子構造(図5.10(a))をもち，規模の異なる6個のデータで，その規模はノード数がそれぞれ36，64，100，144，196，256である。各データ中の配水池，取水点，需要点の個数は全ノード数のそれぞれ10%，10%，40%とした。ポンプは各配水池の出口にあるとした。また，ポンプ圧，標高，取水量，需要量，各管路の抵抗を表す係数も現実の管網に見合う範囲内で乱数を用いて割り当てた。

② データ2

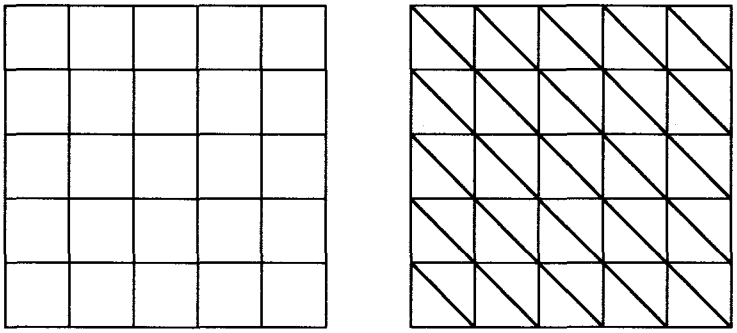
データ 1 の各格子に枝を 1 本追加したネットワークである(図 5.10(b))。データは 3 個でその規模はノード数がそれぞれ 49，100，144 である。その他の条件はデータ 1 と同様である。

③ データ 3

現実の管網データでその規模はノード数 155，管路数 193 である。

④ データ 4

大規模な管網を想定したテストデータで，その規模はノード数 490，管路数 680 である。



(a)　　(b)

図 5.10 格子状ネットワーク

(2) 実験 1 の結果

本文で提案したアルゴリズムの解は非線形関数の折れ線近似による近似誤差を含んでいる。ここでは，解の誤差として KVL を満たした場合の(各ノードにおける)KCL の誤差の最大値を考える。誤差を減らすためには折れ線近似の区分点を増やす必要があるが，それに伴って繰り返し計算回数が増大する。そこで，前記のアルゴリズムでは折れ線近似の精度を徐々に上げることでその増大をおさえようとしている。実験 1 ではその効果を調べるため，精度を固定して解いた場合と，精度を徐々に上げた場合の繰り返し回数，計算時間，解の誤差の比較を行なった。

表 5.1　実験 1 の結果

データ		#1			#2			#3		
ノード数	枝数	回数	時間 (秒)	誤差	回数	時間 (秒)	誤差	回数	時間 (秒)	誤差
36	60	32	0.13	9.01	73	0.25	2.17	14	0.11	0.03
64	116	61	0.31	5.43	143	0.70	2.13	24	0.25	0.03
100	180	93	0.75	8.07	230	1.74	2.06	14	0.33	0.03
144	264	133	1.53	7.91	319	3.59	2.84	31	0.71	0.03
196	364	184	2.96	9.09	467	7.50	2.62	28	0.95	0.03
256	480	256	6.14	9.57	622	11.54	2.35	31	1.58	0.04

表 5.1 にデータ 1 の各管網に対する解析結果を示す。#1，#2 は精度を固定した場合，#3 は精度を 10 回にわたって更新した場合である。なお計算時間の単位は秒，回数は前記アルゴリズムの STEP6 を実行した回数，すなわち，精度を更新した回数と各精度のもとで解をもとめる際，枝電圧が近似区間をまたがって移動した回数の和である。表 5.1 によると精度を徐々に上げながら解いた場合，精度を固定して解いた場合に比べ，繰り返し計算回数，計算時間解の誤差ともはるかに小さくなっていることがわかる。なお，近似区間を縮小して解いた場合，各枝にかかる電圧の絶対値がその初期値の絶対値よりも小さくなる傾向があった。そのため，実験では#3 の場合，折れ線近似の区分点位置を決定するパラメータ r として 14 前後の値を用いた。また，この時 480 枝の例では枝電圧が近似区間をまたがって移動した回数は 1 つの精度あたり約 2 回であった。すなわち，ほとんど(99.5%以上)の場合，解の枝電圧は最初の近似区間におさまっていたことがわかった。

(3)実験 2 の結果

折れ線近似の区分点位置を決定するパラメータ r の影響を確かめるため実験 1 の#3 で使用したパラメータのうち r を 10 に変更してデータ 1 の 256 ノードの管網を解析した。その結果，繰り返し計算回数は 249 回となった。実験 1 の#3 ではこの回数は 31 回であるから区間幅が同じ場合でもその位置が繰り返し計算回数に大きな影響を与えていることがわかる。

表 5.2　実験 2 の結果

データ			折れ線近似手法 +Newton 法		Newton 法	
種類	ノード数	枝数	回数	時間	回数	時間
データ 1	36	60	15	0.12	29	0.30
	64	116	25	0.26	26	0.48
	100	180	15	0.38	33	1.02
	144	264	33	0.82	36	1.72
	196	364	29	1.06	50	3.30
	256	480	32	1.75	43	4.63
データ 2	49	120	12	0.23	23	0.45
	100	261	14	0.55	32	1.43
	144	385	27	1.05	27	1.90
データ 3	155	193	23	0.37	50	1.14
データ 4	490	680	49	2.54	50	6.86

*50 回の反復計算では収束しない

(4)実験 3

実験 1 の#3 で得られた解の誤差は実用上全く問題のない範囲内におさまっているが，この実験では厳密解を求めることを考え，Newton 法を単独で適用した場合と本文で提案した折れ線近似手法の解を初期解として Newton 法を適用した場合の比較を行なった。なお，Newton 法の停止条件は解の誤差が 0.01 以下または繰り返し回数が 50 回に達することとした。実験 3 の結果を表 5.2 に示す。これによると折れ線近似手法と Newton 法を組み合わせた場合，計算時間は Newton 法単独の場合の 1/2 以下となり，しかも全ての例で収束していることがわかる。折れ線近似手法と Newton 法を組み合わせた手法は折れ線近似手法により精度の良い近似解を求めた後 Newton 法を適用するので Newton 法の特徴である解の近傍での収束の速さが生かされ，その収束もほぼ確実である。従って，厳密解を求めるには非常に有効であると言える。

また，本文ではポインタの操作を効率化した LU 分解法を提案したが，この方法の処理時間が，データ領域の設定をあらかじめ行なわない場合の Gauss 法の処理時間の 1/3 以下になることが確認された。なお，実験 3 で用いた Newton 法のプログラムはこの LU 分解法を採用している。

(4)他の手法との比較

大規模な管網の解析を行なった結果については，Newton 法によるもの[30]と数理計画法によるもの[31]が報告されており，いずれも本文に実験で使用した計算機とほぼ同じ性能の計算機で実行している。それによると，Newton 法の場合，ノード数 191，管路数 322 の例で計算時間は約 50 秒である。これに対して本文で提案した手法(折れ線近似手法+Newton 法)では，ノード数 196，管路数 364 の例で計算時間は約 1.1 秒である。また数理計画法による方法ではノード数が約 500 の管網解析の近似解をもとめるのに約 50 秒を要しているが，本文の手法ではノード数 490，管路数 680 の管網解析の厳密解を 2.6 秒で求めることができる。以上の比較から，本文の手法は既存の手法に比べ極めて高速であるといえる。

5.6 結言

本章では Katzenelson 法を改良した折れ線近似手法による管網解析手法を提案した。本手法は管路の非線形特性に注目し，折れ線近似の区分点を適切に選ぶことにより繰り返し計算回数をおさえている点，行列の LU 分解を工夫し処理の高速化を実現している点に特徴がある。その結果，各精度における解の枝電圧はほとんどの場合最初の近似区間内に入っていた。そして，計算時間はこれまでに提案されている手法に比べて約 1/20 となった。しかも，折れ線近似手法は収束性が保証されているため本手法によって近似解を求め，それを初期解として Newton 法を適用することで厳密解をほとんど確実に求めることができる。

本手法は管網解析を目的として開発されたが，管路の非線形特性を変更することにより電気回路解析等の分野への適用も可能である。またその高速性を利用して，計画問題への応用も期待される。

第6章 結論

本研究は，ネットワーク構造を有する組合せおよび数理計画問題に対するアルゴリズムの研究開発を目的としたものである。特に，現実に存在する多くの組合せ問題の内 VLSI・CAD 問題，水道網解析・計画問題および通信網設計の問題をとりあげ，理論と実際の両局面から解決策を導き出すことを試みた。ここでは本研究により得られた成果を要約するとともに今後の課題について述べる。

本論文により得られた成果は以下の通りである。

(1) VLSI のレイアウトにおいては，その代表的問題の一つであるチャンネル配線問題に対して2種類の新しい手法を提案し，計算機実験によりほとんどの場合最適解を短時間で得ることが出来ることを確認した。(2 章)

(2) VLSI レイアウトコンパクション問題に対してグラフ理論とその応用の面から検討を加え，グラフ上の木の初等変換による解法および最小コストフローによる解法を提案した。(3 章)。

(3) LSI 論理設計関係では論理自動合成問題に対して，知識処理とアルゴリズムの組合せによる解法を提案し，既存コンピュータの回路を用いて評価した。その結果熟練技術者と同等の回路を生成できることを確認した。(4 章)

(4) 水道網の流量圧力解析算問題に対して区分線形近似手法に基づく解析手法を提案し，従来の手法に比べ処理速度が 10 倍以上高速であることを確認した。(5 章)

本研究では種々の組合せ問題，数理計画問題を扱い，その解法を提案したが，これらの手法において共通する点がいくつかあげられる。その一つはこれらの問題に対して「グラフ・ネットワークの手法を応用したこと」である。例えば，チャンネル配線問題に対しては，最長経路，最短経路，最小コストマッチング，グラフ中の閉路の探索などのアルゴリズムを使用した。他の問題に対しても，上記の他，最大フロー，最小コストフロー，木の初等変換，最短経路などのアルゴリズムを適用した。さらに，行列演算においても行列をグラフで表現し処理を行なった。これらの基本アルゴリズムを活用した結果，既存の手法に比べ処理の大幅な高速化をはかることができた。

一方，解の探索においては「無駄な探索範囲を削減すること」が重要である。そのため，論理合成問題では未変換部分のコストを評価することで無駄な探索を排除した結果，一部の解空間の探索だけで解を発見することができ，処理時間の大幅な短縮を達成することができた。

また，直接構成法による近似解法では局所的な判断により解空間を順次限定し最終的に一つの解を発見する方法とも解釈できるが，本研究で試みたのは「解空間の限定を可能な限り遅らせること」である。これは，局所的な判断を遅らせれば遅らせるほど，より正確な判断が出来ることから重要である。例えば，チャンネル配線問題のアルゴリズム#1 では割りあてるトラックの指定を最後まで行なわないようにした。アルゴリズム#2 ではネット併合の可能性をマッチングで表現することによってさらに実際のネットの併合時点を遅らせることに成功した。

これらの手法は処理時間の短縮ならびに得られる解の改善を行う上で重要であり，本研究で扱った問題だけでなく一般の組合せ問題，数理計画問題への応用が期待できる。

今後は，個々の問題に対する解法の改善とあわせて，組合せ論的および数理計画法的ネットワーク算法に関する統一的な最適化手法の考察が重要であると考える。

謝辞

本論文を結ぶにあたり，終始身に余る御指導・御鞭撻を賜わり，さらに日頃より御高配を頂いた大阪大学大学院工学研究科情報システム工学専攻白川功教授に衷心より感謝の意を表します。

本研究をまとめる過程で種々の御指導と御鞭撻を頂いた大阪大学大学院工学研究科情報システム工学専攻藤岡弘教授，村上孝三教授ならびに薦田憲久教授に深く感謝いたします。

また本論文作成にあたり，懇篤なる御指導，御高配を頂いた大阪大学大学院工学研究科情報システム工学専攻西尾章治郎教授ならびに電子情報エネルギー工学専攻岸野文郎教授に深謝いたします。

筆者が大阪大学工学部ならびに大学院工学研究科在学中，多大な御指導・御鞭撻を賜わった尾崎弘教授ならびに樹下行三教授に心からの感謝の意を表します。

本研究に対する有益な御助言・御討論を賜わった早稲田大学大附辰夫教授，中央大学伊理正夫教授，カリフォルニア大学 Ernest S. Kuh 教授，奈良先端科学技術大学院大学 藤原秀雄教授，明治大学山田輝彦教授，中央大学築山修治教授，九州工業大学笹尾勤教授ならびに多くのグラフ理論，ネットワーク計画に関する諸先輩の方々に厚く感謝いたします。

本論文の取りまとめの機会を与えて頂き，御理解ある御配慮，御援助を頂いた日本電気(株)石黒辰雄取締役支配人，後藤裕一支配人，後藤敏 C&C 研究所長，山本昌弘パーソナル C&C 事業企画部長には日頃からの御指導に対する感謝ともども厚く御礼申し上げます。さらに藤田友之システム基礎研究部長はじめ同僚諸氏には有益な御討論をいただいたり，プログラム作成面でご協力いただいた。ここに記して，心から感謝致します。

参考文献

[1] 尾崎弘, 白川功, 翁長健治: “グラフ理論”, コロナ社(1975).

[2] A. Hashimoto and S. Stevens: “Wire Routing by Optimizing Channel Assignment within Large Apertures ”, Proc. 8th Design Automation Workshop, pp. 155–167 (1971).

[3] D. N. Deutsch: “A Dogleg Channel Router ”, Proc. 13th Design Automation Conference, pp. 425–433(1976).

[4] G. Persky, D. N. Deutsch and D. G. Schweikert: “LTX- A System or the Directed Automation Design of LSI Circuits ”, Proc. 13th Design Automation Conference, pp. 399–407(1976).

[5] B. W. Kernighan, D. G. Schweikert and G. Persky: “An Optimum Channel-Routing Algorithm for Polycell Layouts of Integrated Circuits ”, Proc. 10th Design automation Workshop, pp. 50–59(1973)

[6] T. Ohtsuki, H. Mori, E. S. Kuh, T. Kashiwabara and T. Fujisawa: “One-Dimensional Logic Gate Assignment and Interval Graphs ”, IEEE Trans. Circuits and Systems, Vol. CAS-26, No.9, pp. 675–684(1979)

[7] C. G. Lekerkerker and J. Ch. Boland: “Representation of a Finite Graph by Set of Intervals on the Real Line ”, Fund. Mathematics, pp. 54–64(1962).

[8] 後藤敏, 大附辰夫: “グラフ理論におけるシンプレックス法ーパスおよびカットセットの最大最小問題の統一的考察ー”, 電子通信学会論文誌, Vol.57-A, No.11, pp.810–817(1974).

[9] 石川正樹, 松田庸雄: “ビルデイングブロック LSI の自動コンパクション手法”, 電子通信学会技術研究報告 CAS83-209(1984).

[10] S. B. Akers, J. M. Geyer and D. L. Roberts: “IC Mask Layout with a Single Conductor Layer ”, Proc. 7th Design Automation Workshop, pp. 7–16(1970).

[11] M. Y. Hsueh and D. O. Pederson: “Computer Aided Layout of LSI Circuit Building Blocks ”, Proc. International Symposium on Circuits and Systems(ISCAS), pp.474–477(1979).

[12] Y. Z. Liao and C.K.Wong: “An Algorithm to Compact a VLSI Symbolic Layout with Mixed Constraints ”, Proc. 20th Design Automation Conference, pp.107–112(1983).

[13] W. Wolf, R. Mathews, J. Newkirk and R. Dutton: “Two Dimensional Compaction Strategies ”, Proc. International Conference on Computer-Aided Design (ICCAD), pp.90–91(1983).

[14] G. Kedem and H. Watanabe: “Graph Optimization Techniques for IC Layout and Compaction ”, Proc. 20th Design Automation Conference, pp.113–120(1983).

[15] M. Schlag, Y. Z. Liao and C. K. Wong: “An Algorithm for Optimal Two- Dimensional Compaction of VLSI Layouts ”, Proc. International Conference on Computer-Aided Design (ICCAD), pp.88–89(1983).

[16] A. E. Dunlop: “SLIM - The Translation of Symbolic Layout into Mask Data ”, Proc. 17th Design Automation Conference, pp.595–602(1980).

[17] N. Weste: “Virtual Grid Symbolic Layout ”, Proc. 18th Design Automation Conference, pp.225–233(1981).

[18] Y. E. Cho, A. J. Korenjak and D. E. Stockton: "FLOSS: An Approach to Automated Layout for High-Volume Designs ", Proc. 14th Design Automation Conference, pp.138–141(1977).

[19] F. M. Maley: "Compaction with Automatic Jog Introduction ", Proc. Chapel Hill Conference on VLSI, pp.261–283(1985).

[20] W. L. Schiele : "Improved Compaction by Minimized Length of Wires ", Proc. 20th Design Automation Conference, , pp. 121–127(1983).

[21] D.G.Boyer: "Symbolic Layout Compaction Review ", Proc. 22nd Design Automation Conference, pp. 383–389 (1988).

[22] T. J. Kowalski and D. E. Thomas: "The VLSI Design Automation Assistant: Prototype System ", Proc. 20th Design Automation Conference, pp. 479–483(1983).

[23] J. A. Darringer, W. H. Joyner, C. L. Berman and L. Trevillyan: "Logic Synthesis Through Local Transformations ", IBM Journal of Research and Development, Vol. 25, No. 4, pp. 272–280(1981).

[24] J. Dussault, C. C. Liaw and M. M. Tong: "A High Level Synthesis Tool for MOS Chip Design ", Proc. 21st Design Automation Conference, pp. 308–314(1984).

[25] T. Shinsha, T. Kubo, M. Hikosaka, K. Akiyama and K. Ishihara: "POLARIS: Polarity Propagation Algorithm for Combinational Logic Synthesis ", Proc. 21st Design Automation Conference, pp. 322–329(1984).

[26] R. Brayton, E. Detjens and S. Krishna: " Multiple-Level Logic Optimization System ", Proc. International Conference on Computer-Aided Design (ICCAD), pp. 356–359(1986).

[27] T. Sasaki, A. Yamada, S. Kato, T. Nakajima, K. Tomita and N. Nomizu: "MIXS: A Mixed Level Simulator for Large Digital System Logic Verification ", Proc. 17th Design Automation Conference, pp. 626–633(1980).

[28] J. R. Duley and D. L. Dietmeyer: "Translation of a DDL Digital System Specification to Boolean Equations ", IEEE Trans. Computers, Vol. C-18, No. 4, pp.305–312(1969).

[29] D. W. Brown: "A State Machine Synthesizer - SMS ", Proc. 18th Design Automation Conference, pp. 301–305(1981).

[30] 小沢純一郎: "配水管網における末端圧保持問題の解法 ", 電子通信学会技術研究報告, CST76–60(1976).

[31] 宮岡伸一郎, 松本邦顕: "最小費用流計算による管路網解析法 ", 電気学会論文誌(C), Vol.101, No.11, pp. 261–268(1981-11)

[32] J. Katzenelson: "An Algorithm for Solving Nonlinear Resistive Networks ", Bell System Technical Journal,Vol.44, pp.1605–1620(1965).

[33] 土木学会: "数理公式集", P379(昭 46)

[34] S.Goto, T.Ohtuski and T.Yoshimura: "Sparse Matrix Techniques for Shortest Path Problem ", IEEE Trans. Circuits and Systems, CAS-23, Vol.12, pp. 752–758(1976)

[35] Markowitz: "The Elimination form of the Inverse and its Application to Linear Programming ", Management Science, Vol.3, No. 4, pp. 255–269 (April 1957).