大規模強連結システムの構造分析に関する研究

—半順序構造化とその応用—

増田達也

内容梗概

本論文は、筆者が大阪大学工学部電気工学教室において行った研究のうち、大規模強連結システムの構造分析に関する研究をまとめたものである。本論文は8章より成り立っており、内容を大別するとつぎの二つの部分に分けることができる。ひとつは、大規模強連結システムの構造分析において必要不可欠である強連結構造の半順序構造化手法について述べた部分(第3章、第4章)であり、他のひとつは、この手法を種々の実際問題に応用した部分(第5章～第7章)である。

本論文の第1章緒論では、システム構造分析の意義、研究過程などを概説し、本研究の背景、目的およびこの分野における位置付けを明らかにしている。

第2章では、本論文の主題である大規模強連結システムの構造分析について議論するに先立ち、まず従来のシステム構造分析において用いられてきた基礎理論および代表的手法について概説し、従来の手法が大規模強連結システムの構造分析には不十分であることを明らかにしている。

第3章では、大規模強連結システムの構造分析において極めて重要となる強連結構造の半順序構造化問題について予備的考察を行っている。まず最初に、本論文を通じて議論の対象となる強連結構造の基本的概念を数学的に定義し、システムが強連結構造となる原因および構造分析における強連結構造の半順序構造化の意義について述べている。ついで、半順序構造化を行う方法としては、構造内に含まれる枝の一部を除去する方法と節点の一部を除去する方法が考えられることを明らかにし、これら二つの方法による半順序構造化問題を具体的に0-1線形計画問題の形に定式化している。

第4章では、第3章で定式化された強連結構造の半順序構造化問題を緩和法の概念を用いて極めて効率よく解く半順序構造化手法を提案し、そのアルゴリズムを具体的に示している。そして、種々の数値実験を行って、本アルゴリズムを演算効率の面で最も効率よく運用するための幾つかの検討を行っている。

第５章、第６章および第７章では、第４章で開発した強連結構造の半順序構造化手法を種々の実際問題、すなわち一対比較多数決を基礎としたランキング問題(第５章)、シミュレーションにおける計算手順決定問題(第６章)およびビルディングブロック方式ＬＳＩの配線問題(第７章)に応用して、本手法の実用性および有効性を実証している。

第８章結論では、本論文で得られた結果を総括している。

大規模強連結システムの構造分析に関する研究

―半順序構造化とその応用―

目　次

第1章　緒　　論

かつては純粋に技術的または経済的な面のみを考慮して解決してきた問題も、社会の多様化、複雑化に伴って、社会的あるいは環境的な面への考慮なしでは解決できない場合が多くなってきた。エネルギー問題、人口問題、環境問題などがこれに相当する。こういった現代社会の諸問題は、互いに複雑に関連して、いわゆる「問題複合体(problematique)[1]」を形成している。近年、システム工学で取り扱う対象もこういった大規模、複雑、多目的かつあいまいな問題複合体のシステムへと広がってきた。これに伴い、このようなシステムの分析、同定あるいは最適化においては、従来からの手法だけでは対処しきれず、その結果、種々の新しいシステム工学的方法論が開発されるようになってきた。

システム構造分析はそのような方法論の一つであり、複雑なシステムを対象とし、問題の探求、構造の把握、解決策の模索などに対するサポートを目的としている。システム構造分析では、まずシステムの構成要素を選定し、目的に応じて適当な二項関係を設定する。つぎに構成要素を一対比較して得られた直接的情報を入力とし、これに二項関係の推移性から導き出された間接的情報を重ね合わせて出力情報を生成する。そして、この出力情報を秩序正しく整理して、最終的にシステム全体の構造モデルを構築していく。

このシステム構造分析の手法に関する研究は、1970年代半ば頃から多くの研究者により盛んに行なわれるようになってきた。その結果、強連結構造(多くのサイクルを含む構造)をもたないようなシステムの構造分析手法に関しては、バッテル・コロンバス研究所においてWarfieldらによって開発されたＩＳＭ法[2]を皮切りに、現在までにいくつかの有効な手法[3],[4]が提案され、既にさまざまな分野において実用に供されている。

しかし、一方、現実のシステムの中には、多くのサイクルを含み、各構成要素が密接に結合しあって強連結構造となるシステムも数多く見受けられる。このような強連結システムに対しては、上述のいずれの手法を用いて構造分析を試みても、殆どの要素がいくつかの強連結成分に含まれてしまい、これ

らの強連結成分と残りの少数要素の関連が明らかになるだけで、強連結成分内の要素間の関係あるいはシステム全体としての構造関係を理解することが困難である。

そこで、こういった強連結システムの構造分析においては、システムの一部の関連を一時的に無視して、推移律を満たす形に構造を多階層化することによって、強連結構造の中に埋没していた構成要素間の関係およびシステム本来の構造的特徴を明確にすることが重要になってくる。本論文では、総ての構成要素間の二項関係が推移律を満足するように、強連結構造を部分的に変形することを「強連結構造の半順序構造化(partial order structuring of strongly connected structure)[5]」と呼ぶことにする。

さて、強連結構造を半順序構造化するための方法論的研究はこれまであまり行なわれてなく、数理計画的なアプローチとして、ダイナミック・プログラミングを利用するUpadhyeらの方法、[6] Bowmanの方法[7]を一般化した井上らの最小損失除去法、[8] 一方、グラフ理論的なアプローチとして、簡略化ルールを用いたPhoの方法[9]があるのみである。しかし、これらの方法では構成要素数の非常に多い大規模強連結構造を取り扱えないという欠点があった。最近、和多田らはこの欠点を補うためにヒューリスティックな方法を提案した。[10] この方法は計算効率がよく、規模の大きい強連結構造でも半順序構造化が可能である。ただこの方法では、規模が大きく、複雑になるにつれて最適な半順序構造を得る割合が減少する傾向のあることが報告されている。

以上で述べてきたように、大規模強連結システムの構造分析において必要不可欠である、強連結構造の半順序構造化のための実用的かつ有効な手法は未だ確立されていないというのが実状である。したがって、さらにこの方向への努力が続けられるべきであると考えられる。

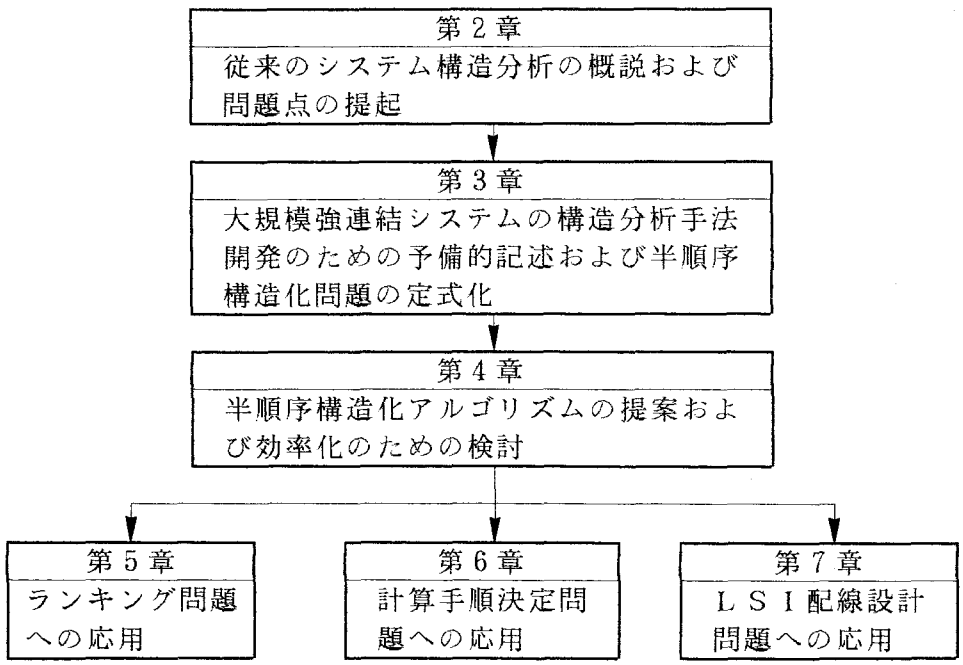
こういった背景のもとに本論文では、大規模で複雑な強連結構造を効率よく半順序構造化することのできる極めて有効な手法を開発し、種々の実際問題へ応用することにより、その実用性を実証することを研究目的としている。本論文で提案する手法においては、強連結構造の半順序構造化問題を構造内

に含まれる総てのサイクルの解消問題と考え、これをシステム構造から抽出されたサイクルを制約式とする0-1線形計画問題に帰着させている。そして、この0-1線形計画問題を解くことにより最適な半順序構造化を行っている。その際、緩和法[11]の概念を導入し、総てのサイクルを抽出する代わりに、少数のサイクルのみを逐次抽出して規模の小さい一連の部分問題のシーケンスを作成し、これを解くことで問題規模の縮小を図っている。したがって、本手法は構成要素数が多く、複雑な大規模強連結システムの構造分析においても十分適用が可能である。

本論文の主要部分は6章から成り立ち、その構成は第1.1図に示すとおりである。以下に、各章の概要を示す。

第2章では、本論文の主題である大規模強連結システムの構造分析について議論するに先立ち、まず従来までのシステム構造分析の概要、その基礎理論および代表的手法について概説し、これらの手法が大規模強連結システムの構造分析には不十分であることを明らかにする。

第3章では、大規模強連結システムの構造分析における半順序構造化問題



第1．1図　論文の構成

およびその意義を明らかにするための予備的な考察を行う。まず最初に、本論文を通じて議論の対象となる強連結構造の基本的概念を定義し、システムが強連結構造となる原因および強連結構造の半順序構造化の意義について述べる。ついで、この半順序構造化問題を二つの方法により、具体的に０－１線形計画問題の形に定式化を行う。

第４章では、第３章で定式化された半順序構造化問題を緩和法の概念を用いて極めて効率よく解く半順序構造化アルゴリズムを提案する。そして、種々の数値実験を行って、本アルゴリズムを特に演算効率の面において、効率よく運用するための幾つかの検討を行う。

第５章、第６章および第７章では、第４章で開発した強連結構造の半順序構造化手法を種々の実際問題、すなわち一対比較多数決を基礎としたランキング問題(第５章)、シミュレーションにおける計算手順決定問題(第６章)およびビルディングブロック方式ＬＳＩの配線問題(第７章)に応用して、システム構造分析における半順序構造化の意義を実用面から明らかにするとともに、本手法の実用性および有効性を実証する。

第８章は本論文の総括である。

第１章の参考文献

（１） The Club of Rome : The Predicament of Mankind, Quest for Structural Responses to Growing World-wide Complexities (1970)

（２） J.N.Warfield : Societal Systems ; Planning, Policy and Complexity, Wiley (1976)

（３） 市川,春名 : Ⅲ. システム技法 第１章 問題発掘と構造化技法, 電気学会誌, Vol.99, No.11, pp.1005-1009 (昭 54)

（４） 戸田,山本 : 構造分析, 計測と制御, Vol.21, No.3, pp.350-357 (昭57)

（５） 増田,藤井 : 緩和法の概念に基づく強連結システムの半順序化とその応用, 計測自動制御学会 第２回知識工学シンポジウム, pp.45-50 (昭59)

応用，計測自動制御学会 第２回知識工学シンポジウム，pp.45-50（昭59）

（６） R.S.Upadhye and E.A.Grens : An Efficient Algorithm for Optimum Decomposition of Recycle Systems, AIChE Journal, Vol.18, No.3, pp.533-539 (1972)

（７） V.J.Bowman and C.S.Colantoni : Majority Rule Under Transitivity Constraints, Management Science, Vol.19, No.9, pp.1029-1041 (1973)

（８） 井上,守安,木村 ： 推移律を満足する集団選好の構造化，計測自動制御学会論文集，Vol.18, No.9, pp.905-911（昭 57）

（９） T.K.Pho and L.Lapidus : Topics in Computer-Aided Design : Part 1. An Optimum Tearing Algorithm for Recycle Systems, AIChE Journal, Vol.19, No.6, pp.1170-1181 (1973)

(10) 和多田,田中,浅居 ： ヒューリスティックな手法による集団選好構造の同定，システムと制御，Vol.26, No.10, pp.671-677（昭 57）

(11) A.M.Geoffrion : Elements of Large Scale Mathematical Programming, Management Science, Vol.16, No.11, pp.652-675 (1970)

第2章　システム構造分析の基礎理論と手法

2．1　緒　言

本章では、本論文の主題である大規模強連結システムの構造分析について議論するに先立ち、まずシステム構造分析の概要、その基礎理論および代表的手法について概説することを目的としている。まず、2.2節ではシステム構造分析とは一体いかなるものか、またその目的は何かといったシステム構造分析の概要について述べる。つぎに2.3節では、構造分析の基礎となる数学的理論、特に二項関係、有向グラフ、二値行列などについて概説する。さらに2.4節では、構造分析の代表的手法として従来よく用いられてきたＩＳＭ法について紹介し、その問題点を明確にする。

2．2　システム構造分析

システムの計画は、目的の明確化、代替案の生成、代替案の評価という3つのフェーズからなっていると考えられる。特に、「システムは、所定の目的を果たすべく、選定され、配列され、連係して動作する一連の構成要素の組み合わせである」[(1)] と言われているように、目的の明確化は、システム開発の基本方針を与えるものである。しかし、これまでのシステム工学の多くの議論は、システムの目的が把握されているという前提で議論が進められてきたといえる。これは、従来のシステムにおいては、比較的少数の目的がシステムを支配しており、それらの抽出が容易で、それらの間の個々の関連を調べることにより全体の目的構造を容易に把握できたためと考えられる。しかし、大規模複雑なシステムにおいては、多様な価値観をもつ多数の関係者が存在し、システムの目的摘出が難しく、しかも個別に関連を調べるだけでは相互関係が複雑で全体の目的体系を把握するのが非常に困難になっている。

同様のシステムの大規模化に伴う全体体系の把握の困難さの問題は、代替案の生成、評価のフェーズにおいても生じている。例えば、システムの機能

を実現するための個々の機能の代替案の数が増え、しかもその相互関係が複雑になり、有望なシステムの代替案の組み合わせを抜けなく抽出することが難しくなっている。また、シミュレーションモデルの構築においても、モデルを構成するパラメータの数が多くなり、まず、パラメータ間の関連の有無を調べ、それらから全体モデルを作り、関係者の合意を得ることが有力となりつつある。さらに、具体的な評価においても、価値体系に関し、関係者の合意を得ることが必要となりつつある。

このように、システム計画においては、詳細な検討に先立ち、全体の構造の大枠をまず把握し、それについて関係者の合意を得ることが重要となっている。この全体の構造把握のためには、多数の要素間の断片的な関連の情報から全体の要素のつながりを整理分析する手法が必要であると考えられる。本論文では、この要素間の断片的な関連の情報をもとに、全体の要素のつながりの様子を明らかにし、システム全体の構造モデルを構築することを「システム構造分析(system structure analysis)」と呼ぶ。

システム工学の分野におけるシステム構造分析に関する方法論的研究は、この十数年来多くの研究者により盛んに行なわれるようになってきた。その結果、比較的規模の小さいシステムに対する構造分析に関しては、十分実用に耐えうる有効な手法が既にいくつか提案されている。[2]〜[6] しかし、膨大な数の要素が密接に関連し合っている大規模強連結システムの構造分析に至っては、未だ実用に耐えうる手法が確立されているとは言い難く、その理論的および方法論的研究が急務とされている。次節以降では、現在までに用いられてきたシステム構造分析の基礎理論およびその代表的な手法について概観することにする。

2．3　構造分析の基礎理論[7]〜[9]

2．3．1　要素の集合と二項関係

対象とするシステムは、1個の要素$v_1, v_2, \cdots, v_l$からなる集合Vであるとする。いま、要素v_iとv_jの順序対(ordered pair)[注2-1]を(v_i, v_j)で表す。集

合Vに対して、集合

$$V \times V = \{(v_i, v_j) : v_i, v_j \in V\} \qquad (2.1)$$

をVの直積集合という。集合Vに属する2つの要素にかかわる関係(注2-2)—これを二項関係(binary relation)という—が1つ規定されると、この関係を満たすような要素の順序対$(v_i, v_j) \in V \times V$全体の集合Rが定まる。そして、V上の二項関係Rとは、直積集合$V \times V$の1つの部分集合を意味し、

$$R \subset V \times V \qquad (2.2)$$

と表せる。$(v_i, v_j) \in R$ならば「v_iはv_jに対してRという関係にある」ことを意味し、$v_i R v_j$と書くことにする。また、$(v_i, v_j) \notin R$ならば「v_iはv_jに対してRという関係にない」ことを意味し、$v_i \underline{R} v_j$と書く。この二項関係の具体的な例としては

優先する、先行する、重要である、好ましい、促進する、抑圧する、
悪化させる、引用する、支持する、報告する、必要とする

など多くのものが考えられる。

V上の二項関係の性質に関して重要なものを以下に挙げておく。

a) 反射律(reflexive law)

$$v_i R v_i, \quad \forall v_i \in V \qquad (2.3)$$

b) 推移律(transitive law)

$$v_i R v_j, \quad v_j R v_k \Rightarrow v_i R v_k, \quad \forall v_i, v_j, v_k \in V \qquad (2.4)$$

c) 対称律(symmetric law)

$$v_i R v_j \Rightarrow v_j R v_i, \quad \forall v_i, v_j \in V \qquad (2.5)$$

(注2-1) (v_i, v_j)と(v_j, v_i)とは$v_i = v_j$の場合を除いて相異なるものと考える。

(注2-2) 例えば、大小関係、前後関係、因果関係、選好関係など。

d) 非対称律(asymmetric law)

$$v_i R v_j \Rightarrow v_j \underline{R} v_i, \quad \forall v_i, v_j \in V \qquad (2.6)$$

e) 反対称律(antisymmetric law)

$$v_i R v_j, \ v_j R v_i \Rightarrow v_i = v_j, \quad \forall v_i, v_j \in V \qquad (2.7)$$

f) 連結律(complete)

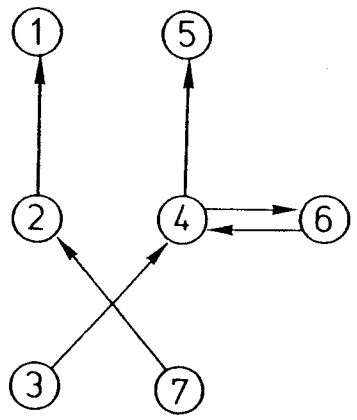
$$v_i R v_j \text{ or } v_j R v_i, \quad \forall v_i, v_j \in V \qquad (2.8)$$

特に、反射的で推移的な二項関係を半順序関係(partial ordering relation)、連結的で推移的な二項関係を弱順序関係(weak ordering relation)、連結的、推移的、対称的な二項関係を全順序関係(total ordering relation)という。また、 反射的、推移的、対称的な二項関係を同値関係(equivalence relation)という。

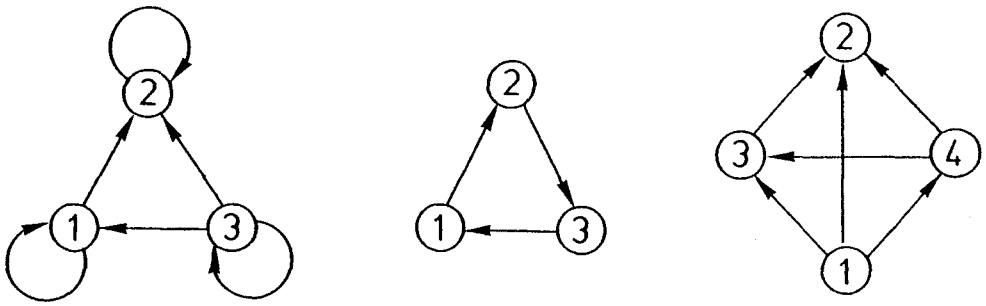
2．3．2 有向グラフと二値行列

いくつかの節点(vertex)とそれらの2点間を結ぶ枝(edge)によって表せる図形をグラフ(graph)といい、中でもすべての枝に向きのついているグラフを有向グラフ(directed graph, digraph)という。

集合V上の二項関係Rは、つぎのようにして有向グラフに表すことができる。Vの要素v_iを有向グラフの節点で表し、$v_i R v_j$のとき節点v_iから節点v_jへ向かう有向枝を描く。有向グラフの一例を第2.1図に示す。すでに述べた二項関係の性質は有向グラフの上につぎのように現れる。反射的な有向グラフとは各節点に1つのループをもったグラフで、その例を第2.2(a)図に示す。また、推移的な有向グラフとは、節点v_iからv_jへ向かう有向枝および節点v_jからv_kへ向かう



第2．1図 有向グラフの一例



(a) 反射的な有向グラフ　(b) 推移的でない有向グラフ　(c) 推移的、非対称的、連結的な有向グラフ

第２．２図　二項関係の性質と有向グラフとの対応

有向枝があるときに、節点v_iからv_kへ向かう有向枝が必ずあるグラフを表す。

第2.2(b)図には推移的でない有向グラフの一例を、また第2.2(c)図には推移的、非対称的で連結的な有向グラフの一例を挙げておく。

集合Ｖ上の二項関係Ｒは、つぎの定義により1×1の二値行列(binary matrix)Ａで表すことができる。

【定義２．１】 次式によって、その要素a_{ij}が定義される二値行列Ａのことを二項関係Ｒの隣接行列(adjacency matrix)と呼ぶ。

$$a_{ij} \triangleq \begin{cases} 1, & v_i R v_j \\ 0, & v_i \underline{R} v_j \end{cases} \qquad (2.9)$$

第2.1図に示した有向グラフの隣接行列は

$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \qquad (2.10)$$

と書ける。

有向グラフにおいて、節点v_iから節点v_jに至る有向枝があるとき、「v_jはv_iから到達可能である」という。また、節点v_iからv_jに至る有向枝の数を順路の長さといい、特別な場合として「各節点は、それ自身から順路の長さ零で到達可能である」ということにする。すでに定義した隣接行列Aは、長さ1の順路による到達可能性を表している。また、1×1の単位行列Iを加えた行列A＋Iは、長さが零または1の順路による到達可能性を表している。第2.1図の例では

$$A+I = \begin{bmatrix} 1&0&0&0&0&0&0 \\ 1&1&0&0&0&0&0 \\ 0&0&1&1&0&0&0 \\ 0&0&0&1&1&1&0 \\ 0&0&0&0&1&0&0 \\ 0&0&0&1&0&1&0 \\ 0&1&0&0&0&0&1 \end{bmatrix} \qquad (2.11)$$

となる。いま、ブール代数演算のもとで(2.11)式の行列を二乗すると、

$$(A+I)^2 = A^2+A+I = \begin{bmatrix} 1&0&0&0&0&0&0 \\ 1&1&0&0&0&0&0 \\ 0&0&1&1&1&1&0 \\ 0&0&0&1&1&1&0 \\ 0&0&0&0&1&0&0 \\ 0&0&0&1&1&1&0 \\ 1&1&0&0&0&0&1 \end{bmatrix} \qquad (2.12)$$

を得るが、この行列は長さが2以下の順路による到達可能性を表している。そして、順次(A＋I)をかけてゆくと、次式で定義される二値行列Mが得られる。

【定義2.2】 次式を満たす1×1の二値行列Mのことを可到達行列(reachability matrix)という。

$$A+I \neq (A+I)^2 \neq \cdots \neq (A+I)^{r-1} = (A+I)^r = M \qquad (2.13)$$

ただし、$r \leqq 1$である。

この可到達行列は任意の長さの順路による到達可能性を表している。いいかえれば、可到達行列Mの(i,j)要素m_{ij}が1ならば節点v_iから節点v_jに向か

う順路が存在し(到達可能)、m_{ij}が0ならば節点v_iから節点v_jへ向かう順路は存在しない(到達できない)ことを表している。第2.1図の有向グラフに対しては

$$A+I \neq (A+I)^2 = (A+I)^3 = M \qquad (2.14)$$

を得る。そして可到達行列は反射的で推移的な二項関係、すなわち半順序関係を表している。

第2.2(b)図に示した有向グラフの隣接行列は

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \qquad (2.15)$$

で表される。この可到達行列は

$$M = (A+I)^2 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad (2.16)$$

となり、すべての要素が1の行列となる。このように、すべての要素が1の行列を普遍行列(universal matrix)と呼ぶ。可到達行列が普遍行列になる有向グラフを強連結な(strongly connected)有向グラフといい、どの節点からどの節点へも到達可能であることを示している。なお、構造が強連結有向グラフで表せられるようなシステムは本論文の研究主題でもあり、これに関しては第3章でより詳しく論ずることにする。

2.4 構造分析の手法：ISM法

システム構造分析の具体的手法としては、J.N.WarfildによるISM法(interpretive structure modeling)[2]のようにシステム構成要素を何らかの二項関係に従って階層化する手法、E.FontellaらによるDEMATEL法(decision making and technological evaluation labolatory)[3]のように要素間の関連の強さに従い項目を整理する手法、M.McLeanらによるSPIN法[4]のようにシステムの動的構造を分析する手法、R.Axelrodらに

よる認知構造図法(cognitive map method)[5]のように符号付有向グラフを使って構造分析を行う手法あるいはR.H.AtkinによるQ-Analysis法[6]のように位相幾何学における複体により構造モデル化する手法などが現在までに提案されている。

これらの中でも、特にＩＳＭ法は最も実用的な手法として各界から注目を浴びている。ＩＳＭ法は大きく分けて、つぎの二つのフェーズによって構成される。

i) 可到達行列の創成フェーズ：人間とコンピュータの対話によってシステム構成要素間の一対比較を行い、これによって人間(集団)の頭の中に陰に存在するメンタルモデルを可到達行列モデルに置き換えるフェーズ。

ii) 可到達行列の分割と抽出フェーズ：この可到達行列モデルをグラフ理論に基づき分割および抽出の操作を施し、スケルトン行列の形に整理し直し、これをもとにメンタルモデルを最終的に多階層有向グラフの形にまとめあげるフェーズ。

以下、本節ではＩＳＭ法を上記の二つのフェーズに従って紹介することにする。

２．４．１　可到達行列の創成[2),(7]

まず、要素集合Ｖに属する任意の要素v_iと、その他の要素$v_j(\neq v_i)\in V$との一対比較を行ったものとして、４つの集合

$$L(v_i) \triangleq \{v_j(\neq v_i)\in V \mid v_i R v_j,\ v_j \underline{R} v_i\} \qquad (2.17)$$

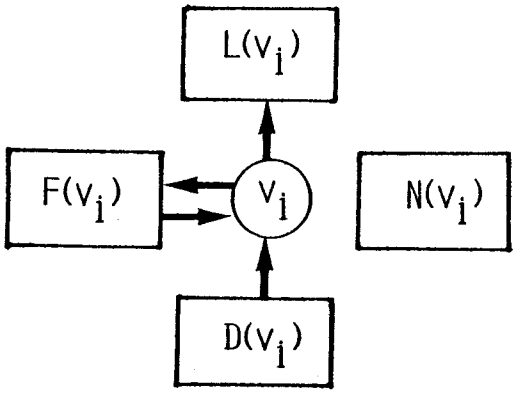
$$F(v_i) \triangleq \{v_j(\neq v_i)\in V \mid v_i R v_j,\ v_j R v_i\} \qquad (2.18)$$

$$N(v_i) \triangleq \{v_j(\neq v_i)\in V \mid v_i \underline{R} v_j,\ v_j \underline{R} v_i\} \qquad (2.19)$$

$$D(v_i) \triangleq \{v_j(\neq v_i)\in V \mid v_i \underline{R} v_j,\ v_j R v_i\} \qquad (2.20)$$

を定義しておく。これらの集合はおのおの

$L(v_i)$：v_iの上位集合(lift set)



第２．３図　有向グラフにおける v_i と $L(v_i)$、$F(v_i)$、$N(v_i)$ および $D(v_i)$ の位置づけ

$F(v_i)$ ： v_i のフィードバック集合(feedback set)

$N(v_i)$ ： v_i の無関連集合(vacancy set)

$D(v_i)$ ： v_i の下位集合(drop set)

と呼ばれ、有向グラフの中では、各々の集合は第2.3図に示すような位置づけになる。そして、v_i を対象にした一連の一対比較の結果に基づいて、創成しようとしている可到達行列Mの(i,j)要素 m_{ij} の値を

$$m_{ij}=1,\ m_{ji}=0,\quad \forall j \text{ such that } v_j \in L(v_i) \qquad (2.21)$$

$$m_{ij}=1,\ m_{ji}=1,\quad \forall j \text{ such that } v_j \in F(v_i) \qquad (2.22)$$

$$m_{ij}=0,\ m_{ji}=0,\quad \forall j \text{ such that } v_j \in N(v_i) \qquad (2.23)$$

$$m_{ij}=0,\ m_{ji}=1,\quad \forall j \text{ such that } v_j \in D(v_i) \qquad (2.24)$$

に設定する。

人間とコンピュータとの対話による可到達行列創成のアルゴリズムは、上に述べた操作の繰り返しのもとで、つぎのように構成される。

Ｓtep １： Ｖから任意に v_i を選び、その他の要素 $v_j(\neq v_i)\in V$ との一対比較を行い、Ｖの中の要素の配列順序を

$$V=\{L_1(v_i), F_1(v_i),\ v_i,\ N_1(v_i), D_1(v_i)\} \qquad (2.25)$$

に並びかえる。これに応じて行列Mの要素を並びかえ、一対比較の終わった行列要素の値を設定する。k＝１とする。ここで、$L_k(\cdot)$ などにおける下指標k は、k回目に構成される$L(\cdot)$などを表す。

Step 2：$L_k(\cdot)$、$N_k(\cdot)$、$D_k(\cdot)$のうち、2個以上の要素を含んでいる各々の集合に対して、その中から任意に1個の要素を選び、その集合に属するその他の要素との一対比較を行い、要素の配列順序を

$$L_k(\cdot)=\{L_{k+1}(v_\alpha),F_{k+1}(v_\alpha),v_\alpha,N_{k+1}(v_\alpha),D_{k+1}(v_\alpha)\} \quad (2.26)$$

$$N_k(\cdot)=\{L_{k+1}(v_\beta),F_{k+1}(v_\beta),v_\beta,N_{k+1}(v_\beta),D_{k+1}(v_\beta)\} \quad (2.27)$$

$$D_k(\cdot)=\{L_{k+1}(v_\gamma),F_{k+1}(v_\gamma),v_\gamma,N_{k+1}(v_\gamma),D_{k+1}(v_\gamma)\} \quad (2.28)$$

に並びかえる。その結果、Vは

$$V=\{\cdots/L_{k+1}(v_\alpha),F_{k+1}(v_\alpha),v_\alpha,N_{k+1}(v_\alpha),D_{k+1}(v_\alpha)/\cdots/F_i(v_i),v_i/\cdots/L_{k+1}(v_\beta),F_{k+1}(v_\beta),v_\beta,N_{k+1}(v_\beta),D_{k+1}(v_\beta)/\cdots/L_{k+1}(v_\gamma),F_{k+1}(v_\gamma),v_\gamma,N_{k+1}(v_\gamma),D_{k+1}(v_\gamma)/\cdots\} \quad (2.29)$$

に並びかわる。この並びかえに応じて行列Mの要素を並びかえ、一対比較の終わった行列要素の値を設定する。k＝k＋1とする。

Step 3：$L_k(\cdot)$、$N_k(\cdot)$、$D_k(\cdot)$に属する要素の数が総て1以下であれば終了。そうでなければStep 2へ戻る。

ここで、対象にしている二項関係Rが推移律を満たすものと仮定できる場合には、Step 1およびStep 2において、推移律によって推定できる行列要素の値は、一対比較を行うことなく推定値を設定する。しかし、上記のアルゴリズムによって創成された行列Mには、推移律によってはその値を推定することのできない相互関連行列(interconnection matrix)と呼ばれる部分が残る。この相互関連行列の部分に関しては、コンピュータが人間に質問を繰り返すことにより、その値を決める。(10)

このようにして、行列Mの総ての要素の値が求まり、このときの行列Mは可到達行列となっている。

２．４．２　可到達行列の分割と抽出[2),(7)]

可到達行列Ｍが創成されると、つぎにＩＳＭ法では行列Ｍの分割と抽出の操作が行なわれる。この操作は４つの過程から構成される。

（１）　パート分割(ブロック対角化)

（２）　レベル分割

（３）　レベル内の分割

（４）　スケルトン行列(skeleton matrix)の抽出

パート分割は、各パートの要素が他のパートの要素と無関係になるように分割することを意味し、行列Ｍをブロック対角化する操作を行う。レベル分割は、１つのパートに属している要素の集合を、各レベル内の要素間の相互関係が対称的($v_i R v_j$ならば$v_j R v_i$)になるように分割することを意味し、ブロック対角化された行列Ｍの各対角ブロックをさらにブロック下三角化する操作を行う。レベル内の分割は、１つのレベルに属している要素の集合を、相互に関係しあうものと無関係なものに分割することを意味し、ブロック下三角化された行列の各対角ブロック(これは対称行列になっている)を再びブロック対角化する操作を行う。このレベル分割およびレベル内分割の結果、各レベルに属している要素のうちの少なくとも１つは他のレベルの要素と上下関係をなし、１つのパートに属している要素集合は、最上位レベルから最下位レベルに順序づけられる。最後に、スケルトン行列の抽出は(１)～(３)でパート分割およびレベル分割された要素集合の二項関係を有向グラフで表現する際に、有向枝の数が最少になるように整理された二値行列すなわちスケルトン行列を求めることを意味する。これら４つの過程を実行するための具体的なアルゴリズムに関しては文献(２)あるいは文献(７)に譲り、ここでは省略する。

以上の４つの過程が終了してスケルトン行列が抽出されると、最後にＩＳＭ法では、このスケルトン行列をもとにして多階層有向グラフ(hierarchical directed graph)が作図される。なお、スケルトン行列から多階層有向グラフを効率よく自動作図する技法も既にいくつか提案されている。[11),(12)]

２．４．３　ＩＳＭ法の問題点

以上、システム構造分析の代表的手法であるＩＳＭ法の概要を紹介した。この汎用プログラムは比較的容易に作製することができるし、一旦プログラムができあがってしまえば、使用者に何ら数学的に高度な知識を要求することなく使えるので、種々の問題に手軽に応用することができる。現在までに、米国を中心に既に50以上の適用例が報告されており、[13],[14] また、方法論的な応用も行なわれている。[15]

しかし、この方法にも以下に示すようないくつかの構造分析上の問題点が残されている。

（１）　ＩＳＭ法では、システム構成要素間の関連の強さを無視して、その有無しか考慮に入れていないために定性的な構造分析しかできなく、定量的な分析は不可能である。

（２）　また、この方法ではサイクル(循環構造)全体を一つのかたまりとみなして構造分析を行うため、強連結構造をもつシステムのように殆どの要素がサイクル内の要素のかたまりに含まれる場合、このかたまりとその他の少数の要素の関連しか整理できなくなり全体の構造分析を行うことができない。また、サイクル内部の関連も整理できない。

このような問題点は、特に強連結システムの構造分析において頻繁に生じる。最近、これらの問題点を克服するためにＩＳＭ法に改良を加えた幾つかの手法が提案され始めている。[16]～[19] しかし、これらの手法はいずれも上記二つの問題点のいずれか一方のみを克服したもので、両方を克服したものではない。また、実用性の面での検証もまだ十分になされていない。したがって、今後は上記の二つの問題点を同時に解決するような手法の開発が、特に大規模強連結システムの構造分析においては非常に重要になるものと考えられる。

２．５　結　言

本章では、システム構造分析の概要を述べたうえで、従来の構造分析にお

いて用いられてきた基礎理論および代表的手法について概説した。

すでに見てきたように、強連結構造をもたないシステムの構造分析に関しては、これまでに多くの手法が提案されており、これらの内のあるものは既に実用に供され、大きな成果を収めている。しかし、一方、膨大な数の要素が密接に関連し合ってサイクルを構成するような大規模強連結システムの構造分析に至っては、既存の手法では幾多の問題点があり、未だ実用に耐えうる手法が確立されているとは言い難い。この面での、有効な構造分析手法の開発が急務とされている。そこで第3章以降では、大規模強連結システムの構造分析に的を絞って、議論を進めることにする。

第2章の参考文献

(1) 猪瀬 : システム工学(I), 岩波書店 (昭 44)

(2) J.N.Warfield : Societal Systems—Planning, Policy and Complexity, pp.208-306 (1976)

(3) H.Thiemann and A.Gabus : 現在社会の諸問題へのシステムズ・アプローチ—DEMATELプロジェクト, ローマ・クラブ東京シンポジウム「新しい世界像を求めて」, pp.21-43, ダイヤモンド社 (昭 49)

(4) M.McLean, et al. : The Importance of Model Structure, Future, pp.40-51 (1976)

(5) R.Axelrod, et al. : Structure of Decision : The Cogitive Maps of Political Elites, Princeton Univ. Press (1976)
あるいは、山本,谷 : 認知構造図, オペレーションズリサーチ, Vol. 24, No.8, pp.462-470 (昭 54)

(6) R.H.Atkin : Combinatorial Connectivities in Social Systems, Birkhauser (1977)

(7) 田村 : 構造モデリング—理論とアルゴリズムを中心にして—, 計測と制御, Vol.18, No.2, pp.170-179 (昭 54)

(8) 片山 : Interpretive Structural Modeling(ISM)手法について, 日本

自動制御協会 多目的システム研究分科会資料, No.77-2, pp.1-10 (昭52)

(9) A.P.Sage : Methodology for Large-Scale Systems, pp.91-164, McGraw-Hill (1977)

(10) J.N.Warfield : Developing Interconnection Matrices in Structural Modeling, IEEE Trans. on Systems, Man and Cybernetics, Vol. SMC-4, No.1, pp.81-87 (1974)

(11) J.N.Warfield : Crossing Theory and Hierarchy Mapping, IEEE Trans. on Systems, Man and Cybernetics, Vol.SMC-7, No.7, pp.505-523 (1977)

(12) K.Sugiyama, et al. : Methods for Visual Understanding of Hierarchical System Structures, IEEE Trans. on Systems, Man and Cybernetics, Vol.SMC-11, No.2, pp.109-125 (1981)

(13) J.N.Warfield : Interpretive Structural Modeling, Notes for Short Course at IEEE 1980 ICCS, Cambridge, Mass. (1980)

(14) D.W.Malone : An Introduction to the Application of Interpretive Structural Modeling, Proc. IEEE, Vol.63, No.3, pp.397-404 (1975)

(15) 田村,楢井 : ＩＳＭによる多目的システムの選好構造把握;ＩＰＳＭ, 計測自動制御学会論文集, Vol.15, No.3, pp.360-365 (昭 54)

(16) 駒谷,福田 : 二項関係の強さを考慮した構造化手法—Revised ISM—, 電気学会論文誌Ｃ, Vol.101, No.5, pp.113-120 (昭 56)

(17) M.Toda, et al. : An Ordering Method for Collective Preference based on a Majority Decision Rule, Proc. IEEE 1980 ICCS, pp.242-247 (1980)

(18) 薦田,春名,中尾 : サイクルを含むシステムの階層的構造分析法—ＨＳＡ, 電気学会論文誌Ｃ, Vol.100, No.12, pp.395-401 (昭 55)

(19) 増田,平峰,藤井 : パラメトリックＩＳＭ法による集団選好の構造化, 第27回システムと制御研究発表講演会講演論文集, W7 (昭 57)

第3章　強連結システムの半順序構造化問題

3．1　緒　言

本章では、本論文で取り扱う問題およびその意義を明らかにするための予備的な記述を目的としている。まず 3.2節では、本論文を通じて議論の対象とする強連結構造の基本的概念を定義し、システムが強連結構造になる原因を明らかにする。つぎに 3.3節では、こういった強連結構造をもつシステムの構造分析においては、構造の一部の関連を一時的に無視して推移律を満たすように半順序構造化すると構造分析上、どのようなメリットがあるか、すなわち半順序構造化の意義を明らかにする。続いて、この半順序構造化の方法としては、枝除去による方法と節点除去による方法の二つが考えられることを述べる。そうして 3.4節では、これら二つの方法に基づいて半順序構造化問題を0－1線形計画問題の形に定式化することを試みる。

3．2　強連結構造[1]

構成要素$v_i \in V \triangleq \{v_1, v_2, \cdots, v_l\}$が何らかの因果関係を意味する$V \times V$上の二項関係Rによって規定されているシステムを考える。2.3.1 で述べたように、一般的にこういったシステムの構造表現には、各構成要素v_iを節点で、二項関係$v_i R v_j$を節点v_iから節点v_jへ向かう有向枝$x_k \triangleq (v_i, v_j) \in X \triangleq \{x_1, x_2, \cdots, x_n\}$で表す有向グラフ$G = \{V, X\}$が用いられる。ここで、VXはそれぞれ有向グラフGの節点集合および有向枝集合であり、l，n は節点および有向枝の総数である。以下では自己サイクルを含まない有限な有向グラフのみを対象とする。また、有向グラフ、有向枝を単にグラフおよび枝と呼ぶことにする。

【定義3．1】 グラフGの枝の順序列 $s \triangleq \{x_1, x_2, \cdots, x_k\}$、$k \leq n$ が

$$x_i = (v_i, v_{i+1}), \quad (1 \leq i \leq k) \qquad (3．1)$$

$$v_i \neq v_j, \quad (1 \leq i < j \leq k+1) \qquad (3．2)$$

を満たすとき、s のなすGの部分グラフ(subgraph)(注3-1) をv_1からv_{k+1}に至る長さk の順路(path)という。また、(3.1)式を満たす s が

$$v_1 = v_{k+1},\quad v_i \neq v_j,\qquad (2 \leq i < j \leq k+1) \qquad (3.3)$$

を満たすとき、s のなす部分グラフを長さk のサイクル(cycle)という。

【定義３．２】 グラフGの任意の２節点 v_i, v_jに対して、v_iからv_jに至る順路およびv_jからv_iに至る順路の両方が存在するとき、このGの構造は強連結構造(strongly connected structure)であるという。また、Gの極大な強連結部分グラフのそれぞれをGの強連結成分(strongly connected component)という。

システムを表現したグラフが強連結になる原因は、Ｖ×Ｖ上の二項関係Ｒにおいて、部分的に推移律

$$v_i R v_j,\ v_j R v_k \Rightarrow v_i R v_k,\quad \forall v_i, v_j, v_k \in V \qquad (3.4)$$

が崩れること、すなわち非推移関係が発生することに起因している。この非推移関係の発生としては、つぎの二つの場合が考えられる。

１） 二項関係Ｒに基づいて構成要素間の関連付けを行う際の、人間の判断誤差に起因する関連付けの誤りによる場合

２） 対象とするシステム自体が本来推移律を満足しないフィードバック構造としての性質をもつ場合

上記のいずれの発生ケースにしろ、一旦システム構造内に非推移関係が発生すると、これらの非推移性によって大小多くのサイクルが発生し、これらが複雑に重なりあって結果的に強連結構造になるものと考えられる。

定義３．２より明らかなように、強連結構造を表すグラフの可到達行列はすべての要素が１である普遍行列となる。その結果、ＩＳＭ法(2) を適用し

(注3-1) s のなす部分グラフとは、グラフ〔$(v_1, \cdots, v_{k+1}), (x_1, \cdots, x_k)$〕を指す。

て構造の階層化を試みてもすべての要素が同一レベルに含まれてしまい、十分な構造分析を行うことができないという問題が生じる。

３．３　半順序構造化の意義とその方法

本節では、強連結構造を半順序構造化(partial order structuring)することの意義および半順序構造化の方法について述べる。

前節で述べたようにシステム構造が強連結構造となる場合には十分な構造分析を行うことができない。しかし、こういった強連結構造をもつシステムにおいても、一部の関連を一時的に除去して推移律が満足されるようにその構造を半順序構造化することにより、非推移性の存在する部分、強連結の中に埋没していた要素間の関係およびシステム全体の構造的特徴などを明確にすることができる。その結果、対象とするシステム構造についてより深い理解が得られ、詳細な構造分析を行うことが可能となる。これが強連結システムの構造分析における半順序構造化の意義である。

強連結構造の半順序構造化は、上述したように構造内に存在するすべての非推移関係を除去することであり、これは換言すると構造グラフに含まれるすべてのサイクルを解消することに相当する。サイクルの解消方法としてはつぎの二通りの方法が考えられる。

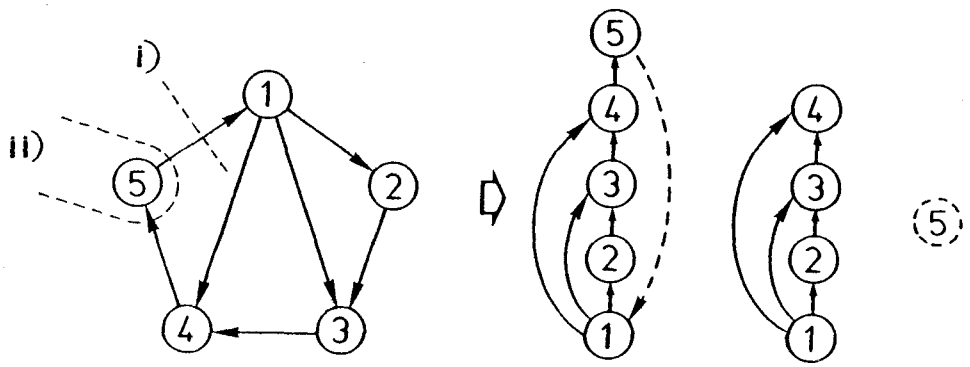
１）　枝除去による方法

２）　節点除去による方法

第3.1図はこれら二つの方法による強連結構造の半順序構造化の一例を示したものである。第3.1(a)図に示す強連結グラフには３個のサイクルが含まれている。これらのサイクルを解消するために、１)の方法を用いて、例えば節点５から節点１に向かう枝(５，１)を除去すると第3.1(b)図のように半順序構造化することができる。一方、２)の方法を用いて、例えば節点５を除去しても第3.1(c)図のように半順序構造化することができる。

次節ではこれらの方法に基づく強連結構造の半順序構造化問題を数学的に

定式化することを考える。



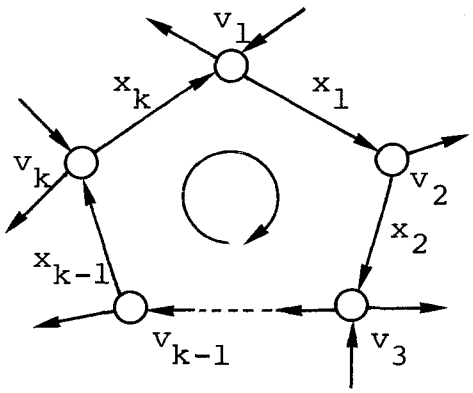
(a) 強連結グラフ　(b) 枝除去による半順序構造化　(c) 節点除去による半順序構造化

第３．１図　強連結構造の半順序構造化の方法

３．４　半順序構造化問題の定式化[3]

３．４．１　枝除去による半順序構造化問題

いま、(3.1)、(3.3)式で与えられる長さk の１つのサイクルを考える(第3.2図参照)。このサイクルが解消されるための必要十分条件は、明らかにサイクルを構成している k 本の枝のうちの少なくとも１本以上の枝が除去されることである。これを各枝x_jを０－１変数と考えて定式化すると、つぎのようになる。



第３．２図　長さk のサイクル

$$x_1 + x_2 + \cdots + x_k \leqq k - 1 \qquad (3.5)$$

$$x_j = 0 \text{ or } 1, \quad (j = 1, 2, \cdots, k) \qquad (3.6)$$

したがって、グラフに含まれる総てのサイクルを解消するためには、枝に対応とする変数$\mathbf{x} = (x_1, x_2, \cdots, x_n)^t \in X$が次式で与えられる制約式を満たさなければならない。

$$\sum_{x_j \in s_i} x_j \leqq k_i - 1, \quad (i = 1, 2, \cdots, m) \qquad (3.7)$$

$$x_j = 0 \text{ or } 1, \quad (j = 1, 2, \cdots, n) \qquad (3.8)$$

ただし、s_i はi番目のサイクルを構成している枝の順序列、k_i はi番目のサイクルの長さ、m はグラフに含まれるサイクルの総数を表す。

つぎに目的関数の定式化を行う。強連結構造を半順序構造化するには、何通りもの枝の除去の仕方が考えられる。例えば、総ての枝を除去しても半順序構造化は可能である。しかし、もとの構造にできるだけ近い形に半順序構造化を行うためには、グラフの既存の枝は除去しない、すなわち$x_j = 1$であるほうがよい。そこで、もしある枝が除去された場合にはその枝に付随したペナルティが課せられるものと考え、目的関数としては半順序構造化に際して余儀なく除去された枝に課かるペナルティの総和を考える。すなわち、

$$f(\mathbf{x}) = \sum_{j=1}^{n} w_j (1 - x_j) \qquad (3.9)$$

ここで 、$w_j (\geqq 0)$は枝x_jが除去された場合($x_j = 0$)に課かるペナルティを表す重み係数である。w_jを総て等しくした場合、(3.9)式は「最少数の枝除去による半順序構造化問題、いわゆる最小帰還枝集合問題(minimum feedback edge set problem)[4]」を意味する。

結局、枝除去による強連結構造の半順序構造化問題は、サイクルの総数 m、枝の総数 n をそれぞれ制約式および変数としてもつ0-1線形計画問題(0-1 linear programming problem)として、つぎのように定式化できる。

[問題Ｐ]

$$\underset{\mathbf{x}\in X}{\text{minimize}} \quad f(\mathbf{x}) = \sum_{j=1}^{n} w_j(1 - x_j) \qquad (3.10)$$

subject to

$$\sum_{x_j \in s_i} x_j \leqq k_i - 1, \quad (i = 1, 2, \cdots, m) \qquad (3.11)$$

$$x_i = 0 \text{ or } 1, \quad (j = 1, 2, \cdots, n) \qquad (3.12)$$

3．4．2　節点除去による半順序構造化問題

つぎに節点除去による半順序構造化問題の定式化を行う。3.4.1の第3.2図のサイクルを節点除去によって解消するための必要十分条件は、枝除去による場合と同様にサイクルを構成している k個の節点のうちの少なくとも１個以上の節点を除去することである(なお、この場合は除去される節点に出入りする枝も同時に除去する)。これを各節点v_jを０－１変数と考えて定式化するとつぎのようになる。

$$v_1 + v_2 + \cdots + v_k \leqq k - 1 \qquad (3.13)$$

$$v_j = 0 \text{ or } 1, \quad (j = 1, 2, \cdots, k) \qquad (3.14)$$

したがって節点除去による半順序構造化問題は、サイクルの総数 m、節点の総数 l をそれぞれ制約式および変数としてもつ０－１線形計画問題として、つぎのように定式化できる。

[問題Ｐ']

$$\underset{\mathbf{v}\in V}{\text{minimize}} \quad f(\mathbf{v}) = \sum_{j=1}^{l} w_j'(1 - v_j) \qquad (3.15)$$

subject to

$$\sum_{v_j \in s_i'} v_j \leqq k_i - 1, \quad (i = 1, 2, \cdots, m) \qquad (3.16)$$

$$v_j = 0 \text{ or } 1, \quad (j = 1, 2, \cdots, l) \qquad (3.17)$$

ただし、w_j' (≥ 0)は節点v_jが除去された場合($v_j = 0$)に課かるペナルティを表す重み係数、s_i'は i番目のサイクルを構成している節点の順序列である。ここで w_j'を総て等しくした場合、[問題P']は「 最少数の節点除去による半順序構造化問題、いわゆる最小帰還節点集合問題(minimum feedback vertex set problem)[4]」を意味する。

なお、(3.11)、(3.16)式を作成するためには、グラフに含まれる総てのサイクルを抽出しなければならないが、これはサイクル列挙法[5]～[7]を用いて容易に行うことができる。これらの列挙法の中でも特に、無駄な探索の繰り返しを防ぐためにblocking-unblocking技法を導入したJohnsonのアルゴリズム[7]は、スペース複雑度(space complexity)が$O(l+n)$、時間複雑度(time complexity)が$O((l+n)(m+1))$であり、[4]従来のアルゴリズムの中で最も効率のよいアルゴリズムであると言われている。

結局、枝除去および節点除去による半順序構造化問題PおよびP′は変数の表す意味が異なるだけで定式化された問題の形自体は同じである。したがって、全く同じ数学的取り扱いが可能である。第3.1表に両者の類似点および相違点を掲げておく。次章以降では、特に断りのない限り枝除去による半順序構造化問題Pを単に問題Pと記述し、話を進めることにする。

第3.1表　半順序構造化問題PとP′の類似点および相違点

		問　題　P	問　題　P′
目的関数の表す意味		除去枝のペナルティ総和の最小化	除去節点のペナルティ総和の最小化
変数	表す意味	枝の除去、非除去	節点の除去、非除去
	個　数	枝の総数　n	節点の総数　l
制約式	表す意味	枝除去によるサイクルの解消	節点除去によるサイクルの解消
	個　数	サイクルの総数　m	サイクルの総数　m

３．５　結　言

本章では、本論文を通じて議論の対象とする強連結システムの半順序構造化問題に関して種々の予備的考察を行った。ここで得られた結果を要約するとつぎの通りである。

（１）　まず初めに強連結構造の基本的概念を数学的に定義し、システム構造が強連結になる原因は構造内に存在する非推移関係(サイクル)にあることを明確にした。

（２）　つぎに、こういった強連結システムの構造分析においては、構造内に存在する総てのサイクルを解消して半順序構造化することが極めて有用であることを明らかにした。すなわち、半順序構造化することにより、強連結の中に埋没していた要素間の関係を浮き彫りにすることができ、より詳細な構造分析が可能となることを示唆した。

（３）　つぎに、強連結構造の具体的な半順序構造化の方法として、枝除去による方法と節点除去による方法の二通りの方法を示した。

（４）　最後に、これら二通りの方法による半順序構造化を数理計画問題として定式化することを試みた。その結果、枝除去による半順序構造化は構造内に含まれる枝およびサイクルをそれぞれ変数および制約式としてもつ０－１線形計画問題に、一方節点除去による半順序構造化は節点およびサイクルをそれぞれ変数および制約式としてもつ０－１線形計画問題に定式化することができた。

第3章の参考文献

（１）　増田,藤井 ： 強連結構造の半順序化，計測自動制御学会 第9回システムシンポジウム講演論文集，pp.297-302（昭 58）

（２）　J.N.Warfield : Societal Systems, John Wiley & Sons (1977)

（３）　増田,新山,藤井 ： 強連結システムの半順序構造化とその応用，システムと制御，Vol.28, No.8（昭 59）

（４）　白川 ： 組合せ問題における計算複雑度解析，システムと制御，Vol.2

0, No.8, pp.395-404 (昭 51)

(5) P.Mateti and N.Deo : On Algorithms for Enumerating All the Circuits of a Graph, SIAM J.Compt., Vol.5, No.1, pp.90-99 (1976)

(6) R.C.Read and R.E.Tarjan : Bounds on Backtrack Algorithms for Listing Cycles, Paths and Sppanning Trees, Networks, Vol.5, No.5, pp.237-252 (1975)

(7) D.B.Johnson : Finding All the Elementary Circuits of a Directed Graph, SIAM J. Comput., Vol.4, No.1, pp.77-84 (1975)

第4章　緩和法を用いた半順序構造化アルゴリズム

4．1　緒　言

前章3.4節で定式化した半順序構造化問題Pは、サイクルの総数m を制約式としてもつ0－1線形計画問題である。対象とする強連結システムが比較的小規模で、かつ簡単な構造をしている場合には、構造内に含まれるサイクルの数も少なく、問題Pは比較的コンパクトである。このような場合には、既存の0－1整数計画法(0-1 integer programming)[1]を用いて容易に問題Pの最適解(最適な半順序構造)を求めることができる。ところが対象とする強連結システムが大規模かつ複雑な場合、換言すると構造の虫食度(sparsity)(注4-1)が小さい場合には、構造内に含まれるサイクルの数が極端に多くなる。その結果、問題Pの制約式の数が膨大なものとなって 実質上求解が困難になるといった問題が生じる。例えば、l ＝１０の完全対称グラフ(注4-2)の場合、その中に含まれるサイクルの総数は $m = \sum_{i=2}^{l} {}_{l}C_{i} \cdot (i-1)! = 1,112,073$である。[2]

そこで本章ではこの問題点を克服するために、緩和法の概念を導入した問題Pの求解アルゴリズムを新たに開発する。本アルゴリズムは、構造内の総てのサイクルを抽出して大規模な問題Pを解く代わりに、少数のサイクルのみを逐次抽出して問題Pに対する一連の小規模な部分問題を作成し、これらを順次解くことにより最終的に、最適な半順序構造を求めるものである。したがって、本アルゴリズムはサイクルの数が極めて多い強連結システムの半順序構造化にも十分適用可能である。以下本章では、まず4.2節で本アルゴ

(注4-1)　本論文では、虫食度を $1 - n/(l^2-1)$ と定義する。

(注4-2)　完全対称グラフ(complete symmetric graph)とは、どの節点$v_i, v_j \in V$に対してもv_iから出てv_jに入る枝およびv_jから出てv_iに入る枝の両方を1個ずつもつグラフのことである。したがって、節点数l の完全対称グラフは $n=l(l-1)$の枝をもち、その虫食度は0である。

リズムの基礎となる緩和法の概要について簡単に述べ、緩和法が問題Pのタイプの解法としてまさに適していることを明らかにする。つぎに4.3節で緩和法の概念を導入した半順序構造化アルゴリズムを具体的に示す。さらに4.4節では本アルゴリズムを特に演算効率の面において、効率よく運用するための幾つかの検討を行う。

4．2　緩和法の概要

まず初めに、アルゴリズムの基礎となる緩和法(relaxation strategy)[3]について簡単に述べる。

数理計画法の解法方略(solution strategy)の一つである緩和法は、制約式の数が非常に多い数理計画問題(mathematical problem)を対象としてGeoffrionによって提唱された。その概要はつぎの通りである。原問題(primal probem ; 本論文の場合は問題Pに相当する)を解く代わりに、制約式のいくつかを緩和し(すなわち一時的に除いておく)、残りを制約式とする部分問題(subproblem)を解く。この部分問題が許容解(feasible solution)を持たなければ、原問題も許容解を持たない。部分問題が許容解を持ち、得られた部分問題の最適解(optimal solution)が緩和した制約式を満たすならば、この最適解は原問題の最適解である。もしそうでなければ、一つあるいはそれ以上の緩和した制約式を部分問題の制約式に加え、手続きを繰り返すというものである。

したがって緩和法は、最適解で等号が成り立つ活性な(active)制約式の数が比較的少ないことが分かっている問題に対しては、規模の小さい部分問題のみを繰り返して解くことにより原問題の最適解が得られることになり、極めて有効である。半順序構造化問題Pはまさにこの種のタイプの問題である。これは強連結構造の最適な半順序構造化が、グラフを構成している枝のうちの半分以下の枝除去によってなされることから容易に推測できる。(注4-3) す

(注4-3)　最適除去枝の数は、完全対称グラフの場合で n／2、虫食度の大きいグラフでは枝の総数の数パーセントである。

なわち、サイクル数m（＞＞ n）がいかに多くても、実際に最適解で活性となる制約式の数は n／２程度あるいはそれ以下と考えられる。

４．３　半順序構造化アルゴリズム[4),(5)]

半順序構造化アルゴリズムの記述にあたり、問題Ｐ(3.10)～(3.12)式の部分問題P^k をつぎのように定式化しておく。

［部分問題P^k ］

$$\underset{\boldsymbol{x}\in X}{\text{minimize}} \quad f(\boldsymbol{x}) = \sum_{j=1}^{n} w_j(1 - x_j) \qquad (4.1)$$

subject to

$$\sum_{x_j \in s_i} x_j \leq k_i - 1, \qquad i \in L^k \qquad (4.2)$$

$$x_j = 0 \text{ or } 1, \qquad (j = 1, 2, \cdots, n) \qquad (4.3)$$

ここで、L^k は問題Ｐの(3.11)式の部分集合である。

このとき、アルゴリズムはつぎのようになる。

Ｓtep １：システム構造を隣接行列A^0の形で表現する。

Ｓtep ２：サイクル列挙法を用いてA^0よりサイクルを q個抽出し、これらのサイクルから作成される制約式集合をL^1とする。また問題Ｐの目的関数の下限値を $\underline{f} = 0$とする。$k = 1$としてＳtep ３に進む。

Ｓtep ３：部分問題P^k を解く。得られたP^k の最適解を$\boldsymbol{x}^k = (x_1^k, x_2^k, \cdots, x_n^k)^t$ とし、$\boldsymbol{x}^k$ に対応する隣接行列をA^k とする。

Ｓtep ４：A^k にサイクルが含まれていなければ、$\boldsymbol{x}^k$ が問題Ｐの最適解であり、Ｓtep ７に進む。さもなければ、Ｓtep ５に進む。

Ｓtep ５：A^k より新たに q個のサイクルを抽出して、これらのサイクルに相当する制約式集合をV^k とする。また、

$$D^k = \{ i \mid \sum_{x_j \in s_i} x_j^k < k_i - 1, \ i \in L^k \} \qquad (4.4)$$

と定義する。ここでD^k は、集合L^k に属する制約式のうち解$\boldsymbol{x}^k$ に

対して不活性な制約式の集合を意味する。

Step 6：$\underline{f} < f(\boldsymbol{x}^k)$ならば$L^{k+1} = L^k \cup V^k - D^k$ かつ $\underline{f} = f(\boldsymbol{x}^k)$とし、$\underline{f} = f(\boldsymbol{x}^k)$ならば$L^{k+1} = L^k \cup V^k$ として、新たな部分問題P^{k+1}の制約式集合L^{k+1}を作成する。$k = k+1$ としてStep 3に戻る。

Step 7：ISM法を用いてA^k を最少枝数の多階層有向グラフの形に整理する。このとき、除去した枝($x_j^k = 0$の枝)はフィードバック枝の形で復元しておく。

第4.1図は以上の半順序構造化アルゴリズムをフローチャートの形で図示したものである。このアルゴリズムにおいてStep 2からStep 6が緩和法の概念を導入した部分に相当する。本アルゴリズムでは、部分問題P^k の最適解$\boldsymbol{x}^k$ が問題Pの最適解になっているか否かの判定を$\boldsymbol{x}^k$ に対応する隣接行列A^k のグラフにサイクルが残っているか否かで行っている。これはつぎの理由に基づく。もしA^k にサイクルが残っていれば、それらのサイクルは問題Pの緩和した制約式のうち$\boldsymbol{x}^k$ が満足しない制約式に相当し、$\boldsymbol{x}^k$ は問題Pの許容解ではなく、したがって最適解でもない。A^k にサイクルが残っていない場合に限り、$\boldsymbol{x}^k$ は緩和した総ての制約式を満足していることになる。4.2節で述べたように、この場合に限り$\boldsymbol{x}^k$ は問題Pの許容解であり、しかも最適解となっている。また、そのときのA^k は最適な半順序構造を表している。

4．4　効率化のための検討

前節で述べたアルゴリズムを効率よく運用するためには、

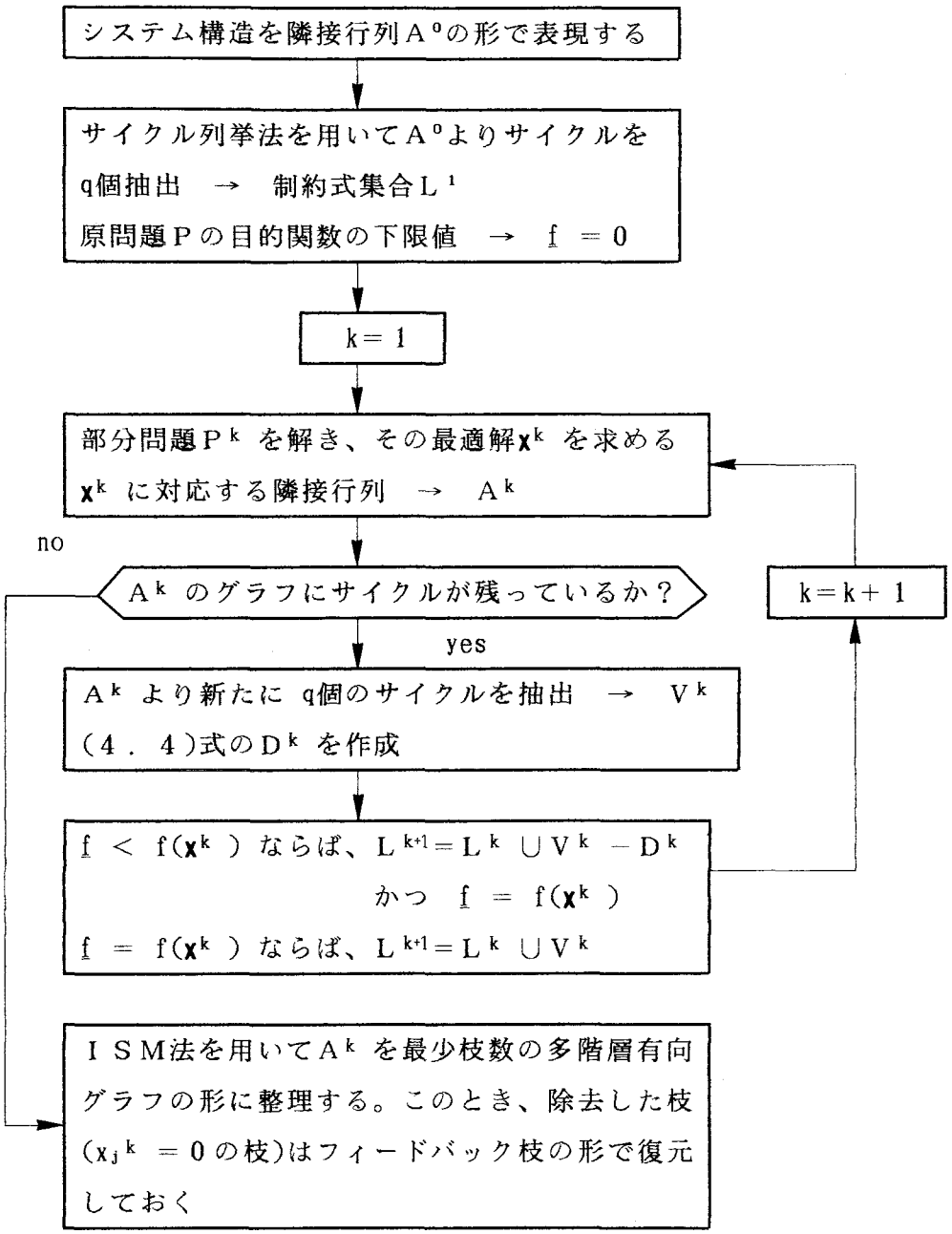
1）部分問題P^k のシーケンスを効率よく解く解法を確立することと、

2）部分問題P^k に新たに追加する制約式の選択個数q を最適な値に設定すること

が重要である。ここで、これらの二点について検討を行う。

4．4．1　部分問題の近似解法(6)

部分問題P^k は変数の総数n が比較的少ない場合($n \leq 50$程度)には、Balas



第４．１図　半順序構造化アルゴリズムのフローチャート

の加法的手法[7]をはじめとする種々の0-1整数計画法を用いて容易に解くことができる。しかし、これらの解法は変数の数が増加するとともに演算時間が指数関数的に増大し、しかも問題のタイプによってその時間が大きく左右される。

この問題点を克服するために、本項では部分問題P^kの特殊性、すなわち

i) (4.2)式の左辺の係数は総て1、またw_j、k_i-1は総て非負、

ii) 制約式の不等式関係は総て「≦」、

iii) $\mathbf{x}=\mathbf{0}$が自明の許容解

であることを利用して、部分問題P^kに適した近似解法のアルゴリズムを示す。

Step 1：自明の許容解$\mathbf{x}^0=\mathbf{0}$を初期暫定解$\hat{\mathbf{x}}$として、$\underline{f}=f(\mathbf{x}^0)=\sum_{j=1}^{n} w_j$とする。また、$s=1$、$I_1^s=\phi$(空集合)とする。

Step 2：i_sをI_1^sに入れ、$\mathbf{b}^0=\mathbf{b}-\mathbf{a}_{i_s}$、$t=1$とする。

Step 3：$i_t \neq i_s$かつ$\mathbf{b}^{t-1}-\mathbf{a}_{i_t}\geqq\mathbf{0}$ならば、$i_t$を$I_1^s$に入れて$\mathbf{b}=\mathbf{b}^{t-1}-\mathbf{a}_{i_t}$とする。さもなければ、$\mathbf{b}^t=\mathbf{b}^{t-1}$として Step 4に進む。

Step 4：$t=t+1$とする。$t\leqq n$ならば Step 3に戻る。さもなければ、$\mathbf{x}(I_1^s)$を求めて、Step 5に進む。

Step 5：$f(\mathbf{x}(I_1^s))<\underline{f}$ならば、$\hat{\mathbf{x}}=\mathbf{x}(I_1^s)$、$\underline{f}=f(\mathbf{x}(I_1^s))$とする。

Step 6：$s=s+1$として、$s>\varepsilon n$ならば$\hat{\mathbf{x}}$を最終的な解と見なして手続きを終了する。$s\leqq\varepsilon n$ならば、$I_1^s=\phi$として Step 2に戻る。

ただし、

$I=\{i_1,i_2,\cdots,i_n\}$：目的関数の係数w_jの大きい順に並び換えた変数x_jの添字集合

I_1：1の値をとる変数の添字集合（$I_1\subseteqq I$）

$\mathbf{x}(I_1)$：集合I_1に属する変数の値を1，属さない変数の値を0として完結した一つの許容解

$\underline{f}$：目的関数の最新の最小値

$\hat{\mathbf{x}}$　：f に対応する暫定解

$\mathbf{a}_t$　：制約式(4.2)式の左辺の係数行列の第t列

$\mathbf{b}$　：制約式(4.2)式の右辺の定数ベクトル（$\mathbf{b}=\mathbf{k}-1$）

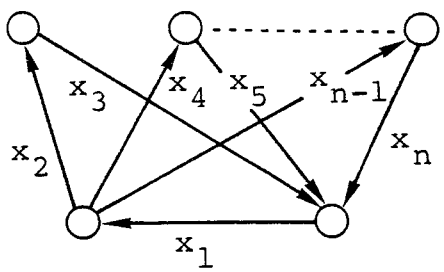
ε　：最適解の探索範囲を限定するパラメータ（$0<\varepsilon\leqq 1$）

s,t　：イテレーション回数を示す指標

である。

上記のアルゴリズムでは、総ての枝を除去した状態に相当する自明の許容解を初期暫定解として選ぶ(Step 1)。続いて一つの変数x_{i_s}を1(x_{i_s}に対応する枝は除去しない)と固定した上で、目的関数値の大幅な改善が期待できるw_jの大きい変数順に1の値を付与し、実行可能な部分解I_1^sのヒューリスティックな生成を逐次行い、一つの完結した許容解$\mathbf{x}(I_1^s)$を得る(Step 2～Step 4)。$\mathbf{x}(I_1^s)$が既に得られている暫定解$\hat{\mathbf{x}}$よりも優れている場合には、これを新たに暫定解とする(Step 5)。以上の操作をw_jの大きい変数順にεn回繰り返しで、最終的に残った暫定解をアルゴリズムで得られる解とする(Step 6)ものである。なお、上記で明らかなようにアルゴリズムの時間複雑度は$O(\varepsilon mn^2)$である。

このアルゴリズムはあくまでも近似解法であり、得られた解が最適解であるという保証はない(本アルゴリズムの解精度に関しては4.4.2で詳しく検討する)。例えば、第4.2図に示すように一本の枝(図中のx_1)に多くのサイクルが集中するグラフの場合、重みw_jの配分比(例えば、$w_1=3$, $w_2=w_3=\cdots=w_n=2$)によっては、いくらεを1に近い値に設定して探索範囲を広げても最適解が得られないことがある。これは、実行可能な部分解I_1^sの生成手順が集合Iに基づいてw_jの大きい変数順



第４．２図　近似解法で最適解が求まらない一例

に行なわれていることに起因している。最適解を得る割合をさらに高めるための一つの手段としては、集合Ｉの順序付けを、w_jの代わりに次式の指標d_jを用いてこの値の大きい順に行い、パラメータθの値をいろいろ変えて本アルゴリズムを繰り返し適用することが考えられる。

$$d_j = \theta \frac{w_j}{\max\limits_j w_j} - (1-\theta) \frac{\sum\limits_{i=1}^{m} a_{ij}}{\max\limits_j \sum\limits_{i=1}^{m} a_{ij}}, \qquad (4.5)$$

$$(j = 1, \cdots, n, \quad 0 \leqq \theta \leqq 1)$$

ただし、a_{ij}は(4.2)式の左辺の係数行列の(i,j)要素である。

(4.5)式の第２項は多くのサイクルに関与する枝は集合Ｉの順序付けにおいて優先されない、すなわちこのような枝はできるだけ除去することを意味している。

４．４．２　近似解法の性能評価[6]

つぎに、この近似解法の性能評価を下記の４つの観点から行う。

i)　どの程度の割合で最適解が得られるか。またパラメータεの設定値がその割合にどのように影響するか。

ii)　最適解が得られなかった場合、近似解法の解はどの程度最適解に近いものか。すなわち、解の精度はどの程度か。

iii)　アルゴリズムの演算効率はどの程度か。

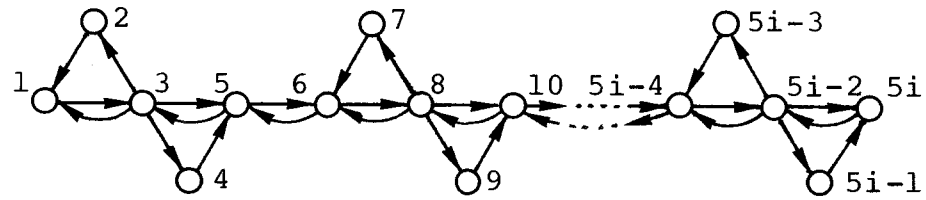
iv)　対象とする強連結グラフの規模によって、演算時間がどの程度必要となるか。

上記の事項を明らかにするために、i)～iii)に関しては規模が異なる50個の部分問題P^kタイプの０－１線形計画問題(問題規模(m,n)＝(5,10)～(160,320))を乱数発生により作成して、実験を行った。また、近似解法で得られた解が最適解であるかどうかを確認するために、これらの問題の最適解をＢalasの加法アルゴリズムを用いて求めた。一方、iv)に関しては、第4.3図に

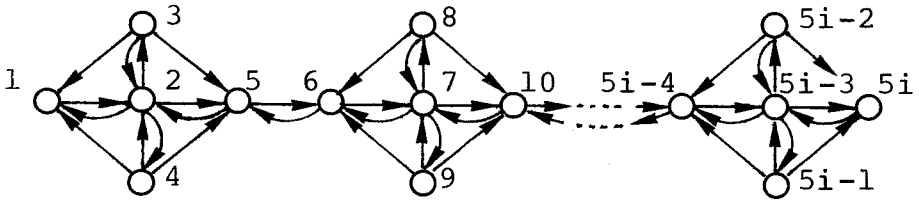
示す3個の規則的な強連結グラフを使用して実験を行った。これらのグラフの節点数l、サイクル数m、枝数n の間にはつぎの関係が成り立っている。ただし、i＝1，2，…である。

$$
\begin{aligned}
&G1: \quad l=5i, \quad m=5i-1, \quad n=10i-2 \\
&G2: \quad l=5i, \quad m=9i-1, \quad n=14i-2 \qquad (4.6) \\
&G3: \quad l=5i, \quad m=35i-1, \quad n=18i-2
\end{aligned}
$$

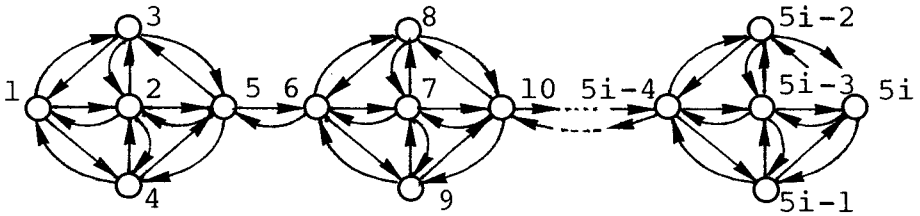
なお、実験はグラフの各枝の重みを総て$w_j=1$、近似解法のパラメータを$\varepsilon=1$に設定して行った。



(a) Gl : l=5i, m=5i-l, n=10i-2



(b) G2 : l=5i, m=9i-l, n=14i-2



(c) G3 : l=5i, m=35i-l, n=18i-2

第4．3図　演算時間の測定に用いた強連結グラフ

結果を第4.1表、第4.2表、第4.3表および第4.4図に示す。第4.1表は、パラメータεの値を変えていったときに最適解が得られる割合がどのように変化するかを示したものである。同表における解の精度は次式のように相対誤差を用いて定義し、表の数値は最適解でなかった解のみの相対誤差の平均値を示している。

$$\frac{|f(\mathbf{x}^*)-f(\hat{\mathbf{x}})|}{|f(\mathbf{x}^*)|} \times 100 \qquad (4.7)$$

ここで、$\mathbf{x}^*$、$\hat{\mathbf{x}}$ はそれぞれ最適解および近似解法で得られた解である。また、第4.2表は50個のうちの比較的規模の小さい5個の問題に対して、近似解法、Balasの加法アルゴリズム(additive algorithm)およびGeoffrionの陰伏的計算法(implicit enumerative method)[8] で解を得るまでに要した演算時間を比較したものである。なお、近似解法の演算時間はε＝1として計算した場合のものである。つぎに第4.4図は規模(m,n)＝(140,280)の問題を例にとって、εの設定値と近似解法の演算時間との関係を示したものである。また、第4.3表は対象とする強連結グラフの問題の規模によって、半順序構造化までに演算時間がどの程度必要かを示したものである。

これらの実験結果から明らかになる事項を以下に要約する。

i) 近似解法によって最適解が得られる割合は、εの設定値を1に近づける、すなわち許容解の探索範囲を広げるほど高くなる。ε＝1.0の場合には、その割合は96%に達している(第4.1表)。(注4-4)

第４．１表　50個の問題に対する近似解法の実験結果

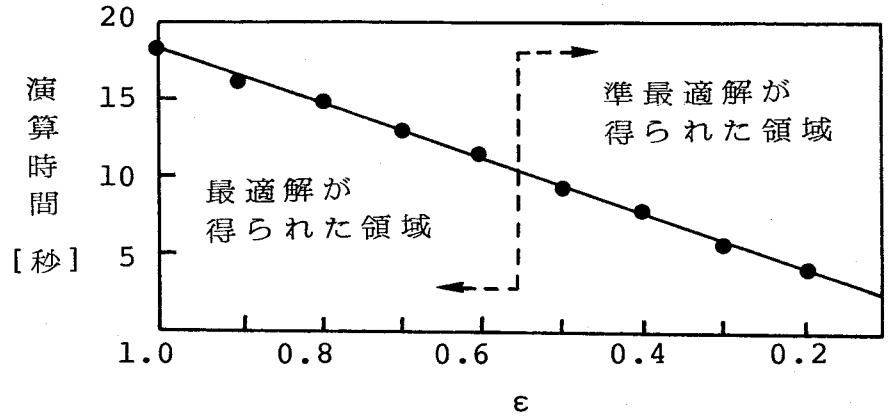
パラメータεの設定値	1.0	0.8	0.6	0.4	0.2	0.1
最適解／準最適解の割合	48/2	46/4	39/11	24/26	15/35	14/36
準最適解の相対誤差の平均値　[%]	2.91	3.28	4.27	6.11	7.64	9.16

第４．２表　近似解法とＢalasおよびＧioffrionのアルゴリズムによる演算時間の比較　［秒］

問題の規模（m,n）	(7,13)	(12,22)	(14,28)	(19,38)	(19,40)
近似解法　：Ａ	0.003	0.012	0.022	0.053	0.058
Ｂalas　：Ｂ	0.011	0.182	0.902	3.302	4.590
Ｇeoffrion：Ｃ	0.007	0.116	0.573	2.096	2.714
比率　Ｂ／Ａ	3.67	15.2	41.0	62.3	79.1
比率　Ｃ／Ａ	2.33	9.67	26.0	39.5	46.8

第４．３表　テストグラフＧ１、Ｇ２、Ｇ３を半順序構造化するのに要した演算時間

	節点数	サイクル数	枝数	演算時間　［秒］		
				サイクルの抽出	0-1線形計画問題の求解	合　計
	l	m	n			
Ｇ１	10	9	18	0.001	0.007	0.008
	20	19	38	0.004	0.051	0.055
	60	59	118	0.032	1.398	1.430
	100	99	198	0.091	6.624	6.715
Ｇ２	10	17	26	0.001	0.022	0.026
	20	35	54	0.004	0.182	0.191
	60	107	166	0.038	5.034	5.072
	100	179	278	0.103	23.497	23.595
Ｇ３	10	69	34	0.005	0.144	0.149
	20	139	70	0.019	1.178	1.197
	60	419	214	0.145	32.161	32.306
	100	699	358	0.388	149.000	149.388



第４．４図　パラメータεと演算時間の関係

ii)　近似解法で得た解が最適解でない場合でも、その解の精度は極めて高く、最適解に非常に近い解(準最適解)を得ることができる(第4.1表)本実験において得た最も精度の悪い解は、規模(m,n)＝(9,17)、ε＝0.1の場合の相対誤差24.9%の解であった。

iii)　近似解法はＢalasの加法アルゴリズムおよびＧeoffrionの陰伏的計算法に比べて計算効率が極めて良く、演算時間のひらきは問題規模が大きくなるにつれて顕著になる(第4.2表)。また、近似解法ではεの設定値を下げることにより演算時間をさらに線形的に短縮することが可能である(第4.4図)。

iv)　本近似解法を用いれば節点数が数百程度の強連結グラフならば、最も多くの演算時間を要するε＝1に設定しても十分実用的な時間内で容易に半順序構造化が可能である。また、所要時間の大部分はサイクル抽出ではなく、０－１線形計画問題を解くのにかかることが分かる(第4.3表)。

(注4-4)　(4.5)式の指標に基づきθ＝1.0および0.8として本近似解法を二度繰り返して適用した場合、この割合をさらに98%まで高めることができた。

以上より、本項で提案した近似解法は部分問題P^kの解法として非常に有力な解法であることが分かる。本近似解法を用いれば短時間で精度の高い解を得ることができ、しかも大規模な強連結システムでも実用的な時間内で半順序構造化することが可能である。

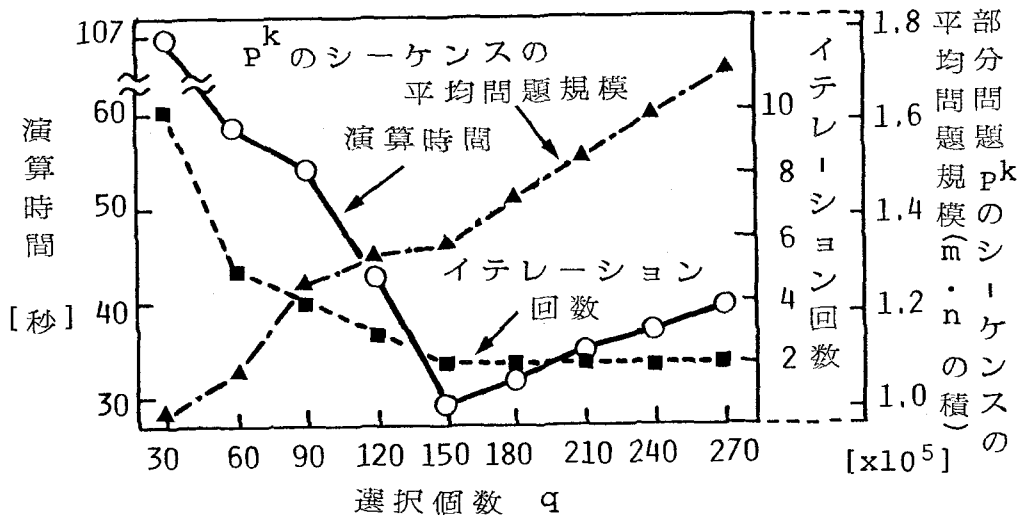
以上のことを総合すると、部分問題P^kの解法としては、結局、変数の数が比較的少ない場合($n \leqq 50$)、換言すれば強連結構造内の枝の数が比較的少ない場合には最適解が確実に得られる従来の0-1整数計画法を、さもない場合($n \geqq 50$)には演算効率の良い上述の近似解法を用いるのが妥当と結論づけることができる。

4．4．3　最適な選択個数の設定目安[9]

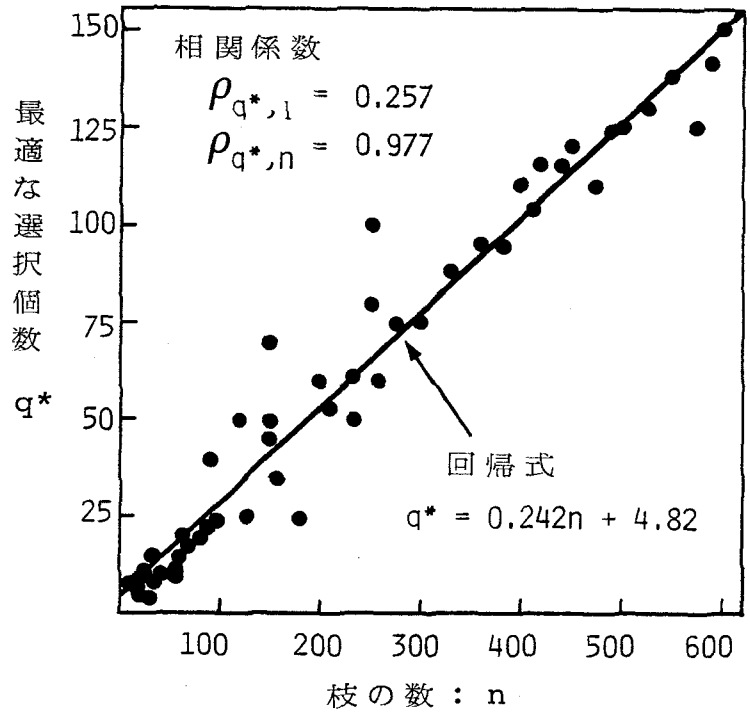
部分問題P^kに逐次追加していく制約式(サイクル)の選択個数qが、アルゴリズムの演算時間に大きく影響する。qを過小な値に設定すれば、各イテレーションでの部分問題P^kの規模は小さく、これを解く時間は少なくてすむが、最適解を得るまでに多くのイテレーションを必要とする。一方、qを過大に設定すれば、これと逆のことがいえる。

第4.5図は上述のことを顕著に示した一例である。同図は、$l=25$、$n=600$の完全対称グラフ(含まれるサイクル数は $m \fallingdotseq 1.76\times10^{24}$)を対象として qをいろいろな値に設定した場合、最適解を得るまでのイテレーション回数、部分問題P^kのシーケンスの平均問題規模および演算時間がどのように変化するかを示したものである。同図から、演算時間を最小にする最適な選択個数q^*が存在することが分かる。このq^*を事前に分かっているグラフの指標 lあるいは nを用いて推定することをつぎに考える。

このために規模(l,n)が(5,10)から(100,600)の50種の強連結グラフを対象に qをいろいろ変えてアルゴリズムを適用し、q^*を求める実験を行った。その結果、第4.6図に示すように、q^*は lよりもむしろ nと高い相関を持つことが判明した。それぞれの相関係数は$\rho_{q^*,l}=0.257$、$\rho_{q^*,n}=0.977$である。そこで、これらの実験データより nを説明変数とするq^*の線形回帰式を求め



第４．５図　制約式の選択個数q の演算時間に与える影響



第４．６図　枝の数n と最適な選択個数q*の関係

た結果、次式を得た。

$$q^* = 0.242n + 4.82 \qquad (4.8)$$

結局、最適な選択個数の設定値は(4.8)式に基づいて、強連結構造内の枝数の約1/4程度にすればよいということができる。

4.5 結　言

本章では、強連結システムの半順序構造化問題を効率よく解くための手法として、緩和法の概念を導入した半順序構造化アルゴリズムを提案した。ここで得られた成果を要約するとつぎの通りである。

(1) まず初めに、極めて多くのサイクルを含む強連結構造の半順序構造化問題を効率よく解くためには、総ての制約式を陽に考慮しなくてもよい緩和法の考え方が有効であることを明らかにした。

(2) つぎに、この緩和法の概念を導入して、強連結構造の半順序構造化アルゴリズムを新たに開発した。これにより、構成要素数が100以上かつ数万のサイクルを含む強連結構造の半順序構造化が極めて容易に行えるようになった。

(3) さらに、上記のアルゴリズムをより効率よく運用するために、部分問題の解法および部分問題に逐次追加する制約式の選択個数に関して検討を行った。その結果、時間複雑度が$O(\varepsilon mn^2)$で、非常に高い割合で最適解を得ることのできる、極めて効率のよい部分問題の近似解法を提案した。また、部分問題に逐次追加していく制約式の選択個数としては、構造内に含まれる枝の数の約1/4程度にその値を設定すると、半順序構造化アルゴリズムの演算効率が最も良くなることを実験的に明らかにした。

第4章の参考文献

(1) 例えば、今野 : 整数計画法, 産業図書 (昭 56)

志水 : システム最適化理論, コロナ社 (昭 51)

(2) 増田,藤井 : 半順序構造化問題の定式化に関する一考察, 計測自動制御学会論文集, Vol.20, No.4, pp.364-366 (昭 59)

(3) A.M.Geoffrion : Elements of Large Scale Mathematical Programming, Management Science, Vol.16, No.11, pp.652-675 (1970)

(4) 増田,新山,藤井 : 緩和法を用いた強連結システムの半順序構造化,第26回自動制御連合講演会, 1066 (昭 58)

(5) 増田,藤井 : 緩和法の概念に基づく強連結システムの半順序化とその応用, 計測自動制御学会 第2回知識工学シンポジウム論文集, pp.45-50 (昭 59)

(6) 増田,新山,藤井 : 強連結システムの半順序構造化とその応用, システムと制御, Vol.28, No.8, pp.544-550 (昭 59)

(7) E.Balas : An Additive Algorithm for Solving Linear Programs with Zero-one Variables, Operations Research, Vol.13, No.4, pp.517-546 (1965)

(8) A.M.Geoffrion : An Improved Implicit Enumeration Approach for Integer Programming, Operations Research, Vol.17, No.3, pp.437-454 (1969)

(9) 増田,藤井 : 強連結構造の半順序化アルゴリズムの開発, 日本自動制御協会 第28回システムと制御研究発表講演会講演論文集, S2 (昭 59)

第5章　一対比較多数決を基礎としたランキング問題への応用

5．1　緒　言

本章および以後の第6章、第7章においては、前章で開発した強連結構造の半順序構造化アルゴリズムを種々の実際問題に応用して、システム構造分析における半順序構造化の意義を実用面から明確にするとともに、アルゴリズムの実用性および有効性を実証することを目的としている。

本章では、最初の応用例として一対比較多数決を基礎としたランキング問題(ranking problem)への適用を試みる。以下に本章の構成を示す。まず、5.2節で一対比較多数決について概説し、一対比較多数決を基礎としたランキング問題における意思決定者集団の選好構造分析に、なぜ強連結構造の半順序構造化が必要になるかを明らかにする。つぎに5.3節では、大阪大学工学部電気工学科に在学する40名の学生を対象にして昭和58年3月9日に実際に行った卒業研究の口頭試問におけるランキング問題を取り上げて、その実験手順および実験結果を示す。さらに5.4節では、この実験結果に半順序構造化アルゴリズムを適用して、この適用によりさらに詳細な順位付けが可能になることを示し、アルゴリズムの特徴およびその有効性を明らかにする。

5．2　一対比較多数決[1]

何人かの意思決定者(decision maker)の集団による意思決定過程(decisin making process)において、各時点で各意思決定者の選好(preference)を集約して、集団の選好を構造化することは、意見の整理、問題点の明確化、少数意見の発掘を促し、合意形成への重要なステップとなる。このような場合、多数決(majority decision)は最も一般的に用いられている。各意思決定者の選好に関する情報を比較的多く、しかも簡便に取り入れる方法としては、多数決を代替案(alternatives)の総ての対に関する一対比較(pairwise comparison)に適用することが考えられる。この一対比較多数決は総ての代替案の一対比較をすることによって、単記投票では失われがちな個々の代替案に

対する選好を序数効用(ordinal utility)として取り入れることができ、集団の選好を集約する際には有用な方法である。しかし、個人の判断誤差により個人の選好が推移律を満足しない場合、あるいは個人の選好は推移律を満足しても、よく知られているように「投票のパラドックス(paradox of voting)[2]」が生じている場合を考えると、一対比較多数決による集団としての選好構造は必ずしも推移律を満足するとは限らない。以下に投票のパラドックスの簡単な例を、k人の意思決定者による集団意思決定問題(group decision making problem)を用いて示す。[3]

いま、代替案の数lは有限であるものとしてその集合を$V=\{v_1,v_2,\cdots,v_l\}$で表し、これらの代替案の総ての一対比較を考える。意思決定者α($\alpha=1,2,\cdots,k$)の選好をV×V上の二項関係R^αで表す。$v_i,v_j\in V$について$v_i R^\alpha v_j$は、意思決定者αにとって「v_iはv_jと同程度か、それ以上に好ましい」ことを表すものとする。

集団としての選好をRで表すことにすると、一対比較多数決は次式で定義されるV×V上の二項関係Rによる選好順序である。

$$v_i R v_j \leftrightarrow \#\{\alpha \mid v_i R^\alpha v_j\} \geqq \#\{\alpha \mid v_j R^\alpha v_i\} \qquad (5.1)$$

ただし、$\#\{\alpha \mid v_i R^\alpha v_j\}$は$v_i R^\alpha v_j$と回答した意思決定者の数を表す。

いま、$v_1,v_2,v_3\in V$について、k=9として

$$\begin{aligned}&\#\{\alpha \mid v_1 R^\alpha v_2 R^\alpha v_3\}=4\\&\#\{\alpha \mid v_2 R^\alpha v_3 R^\alpha v_1\}=3\\&\#\{\alpha \mid v_3 R^\alpha v_1 R^\alpha v_2\}=2\end{aligned} \qquad (5.2)$$

となったとする。このとき、個人の選好は推移律を満足し、しかも線形順序になっている。しかし、各一対比較については

$$\begin{aligned}&\#\{\alpha \mid v_1 R^\alpha v_2\}=6, \quad \#\{\alpha \mid v_2 R^\alpha v_1\}=3\\&\#\{\alpha \mid v_2 R^\alpha v_3\}=7, \quad \#\{\alpha \mid v_3 R^\alpha v_2\}=2\\&\#\{\alpha \mid v_3 R^\alpha v_1\}=5, \quad \#\{\alpha \mid v_1 R^\alpha v_3\}=4\end{aligned} \qquad (5.3)$$

であるから、(5.1)式から単純多数決によれば

$v_1 R v_2$, $v_2 R v_3$, $v_3 R v_1$ （5．4）

と循環順序(サイクル)になり、集団としての選好順序は推移律が崩れる。

一対比較多数決の結果が推移律を満たさなくなるのは、この「投票のパラドックス」と「個人の判断誤差」が複雑に入り交じった結果と考えられる。このような状態は、各意思決定者の意見が互いに異なり、集団としての選好の状況を整理するのが困難なときほど頻繁に発生する。そうして、このような状態においては代替案間の順序関係の多くがサイクルに含まれてしまい、選好構造はいくつものサイクルが重なりあった非常に複雑なものとなる。このままではそれらの代替案に関する意見が錯綜しているという以上には集団としての選好構造を把握することができない。

こういった場合には、もとの多数決の結果をできる限り保存し、しかも推移律を満足する選好構造を求めるために、いわゆる強連結構造の半順序構造化が必要となる。

5．3　口頭試問におけるランキング問題[4),(5)]

本節では、半順序構造化アルゴリズムの適用に先立ち、緒先生方ならびに学生諸氏の御理解と御協力のもとに大阪大学工学部電気工学科において実際に行った一対比較多数決を基礎としたランキング実験およびその実験結果についてまず述べる。ここで取り扱った問題は、大学学部4年生の学生40名(1＝40)を対象とした卒業研究の口頭試問における順位付け問題である。

5．3．1　実験手順

実験はつぎの手順に従って行った。

i)　まず初めに7名の試問委員(k＝7)が各学生の研究発表を聞き、その内容に関していくつかの質問を順次行う。各委員は発表内容および試問結果を十分に考慮した上で、その優劣に関して学生間の一対比較を行い、40名の学生に対する主観的判断に基づく順位付けを

行う。

ii)　つぎに、これら各委員の回答結果を集計し、単純多数決の票数の比率を表す行列、すなわち比率行列(proportion matrix)[3),(6)] M＝(m_{ij})を計算機を用いて作成する。

$$m_{ij} \triangleq \frac{\#\{\alpha \mid v_i R^{\alpha} v_j\}}{\#\{\alpha \mid v_i R^{\alpha} v_j\} + \#\{\alpha \mid v_j R^{\alpha} v_i\}} \qquad (5.5)$$

$$0 \leqq m_{ij} \leqq 1 \qquad (5.6)$$

$$(i,j = 1,2,\cdots,40 \quad \alpha = 1,2,\cdots,7)$$

ここで、一対比較多数決の定義式(5.1)式より

$$m_{ij} \geqq 1/2 \leftrightarrow v_i R v_j \qquad (5.7)$$

が成立し、m_{ij}はその値が1に近いほど委員間の意見の一致度が高いことを表している。一方、$m_{ij}=0.5$は意見が完全に分かれた状態に相当する。

iii)　さらにこの比率行列Mより、単純多数決による集団としての選好関係を示す行列、すなわち多数決行列(majority matrix)[3),(6)] A＝(a_{ij})をつぎの定義により作成する。

$$a_{ij} \triangleq \begin{cases} 1 & \text{if} \quad m_{ij} \geqq 0.5 \\ 0 & \text{if} \quad m_{ij} < 0.5 \end{cases} \qquad (5.8)$$

$$(i,j = 1,2,\cdots,40)$$

5．3．2　実験結果

以上の手順によって作成された行列Aで表される学生間の順位関係を、ISM法を用いて最少枝数の多階層有向グラフの形で表現すると第5.2(a)図のようになる。(注5-1)同図は大きさ29および8の2つの強連結成分S_1、S_2を含

(注5-1)　第5.2図のグラフは、通常の定義と逆に$v_i R v_j$のとき、節点v_jから節点v_iへ枝の向きを定めている。

んでおり、５つのレベルにしか階層化できない。このままではわかりきった順序関係のみが求められたにすぎず、順位付けとしてはあまり意味をなさない。

第5.1図は行列Ａから強連結成分S_1、S_2に対応する(29×29)および(８×８)の小行列W_1、W_2を抜き出したものである。同図では行列Ａにおける$a_{ij}=0$の要素は＊で、一方$a_{ij}=1$の要素はその枝に対応する重みで表してい

i＼j	1	2	3	6	7	9	10	11	12	13	14	16	17	18	19	20	22	26	27	28	29	30	31	34	35	36	37	38	39
1	*	*	*	[0]	*	0	*	*	[0]	*	0	*	*	*	8	*	*	*	20	*	10	20	*	0	0	10	*	8	*
2	25	*	13	40	0	38	40	*	38	[0]	40	40	10	10	40	10	20	30	50	[0]	40	50	40	30	30	40	30	40	10
3	42	*	*	40	17	17	25	[0]	25	*	40	20	25	17	25	*	10	10	50	*	40	50	30	30	40	40	30	33	30
6	0	*	*	*	*	10	8	*	0	*	8	*	*	*	25	0	8	*	33	*	17	42	17	25	17	33	*	25	*
7	25	[0]	*	10	*	17	8	*	33	*	30	[10]	*	*	33	8	30	0	40	*	10	50	40	40	30	50	10	42	10
9	[0]	*	*	*	*	*	[17]	*	[0]	*	*	[0]	*	*	8	*	*	*	0	*	*	20	[0]	0	*	10	*	25	*
10	8	*	*	*	*	*	*	8	*	8	*	*	*	*	29	*	*	*	25	*	17	42	17	25	17	17	[0]	29	*
11	42	13	0	20	17	17	33	*	33	*	40	0	8	0	50	0	20	20	50	*	50	50	40	40	50	50	30	50	30
12	0	*	*	[0]	*	0	*	*	*	*	0	*	*	*	0	*	*	*	10	*	0	30	10	10	0	0	*	0	*
13	50	0	10	50	30	40	50	30	30	*	50	25	50	25	42	17	33	42	50	8	50	50	42	50	50	50	42	42	50
14	[0]	*	*	*	*	10	*	*	[0]	*	*	*	*	*	25	*	*	*	33	*	[0]	25	[0]	8	0	17	*	25	*
16	10	*	*	8	*	0	17	[0]	20	*	8	*	17	*	33	8	8	*	25	*	17	42	33	33	17	33	0	33	0
17	33	*	*	17	17	8	7	*	17	*	17	*	*	[0]	29	*	8	*	42	*	33	42	17	33	42	42	8	29	8
18	33	*	*	33	25	8	14	[0]	25	*	33	17	0	*	29	7	33	17	42	*	33	42	33	33	33	33	25	36	17
19	*	*	*	*	*	*	*	*	[0]	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	[0]	*
20	33	*	[8]	0	*	8	7	[0]	25	*	17	*	[7]	*	43	*	8	[0]	50	*	17	50	25	33	42	42	*	36	[8]
22	20	*	*	*	*	20	8	*	30	*	8	*	*	*	50	*	*	*	33	*	8	33	33	33	17	50	*	42	*
26	20	*	*	25	[0]	20	17	*	10	*	25	[8]	[8]	*	33	0	8	*	50	*	42	50	25	33	33	42	8	33	*
27	*	*	*	*	*	[0]	*	*	*	*	*	*	*	*	17	*	*	*	*	*	*	[0]	*	0	*	*	*	[0]	*
28	33	0	17	33	8	17	50	17	50	*	33	25	29	21	50	21	17	17	33	*	33	50	50	42	33	50	33	50	25
29	*	*	*	*	*	10	*	*	[0]	*	0	*	*	*	25	*	*	*	33	*	*	33	[0]	17	17	25	*	17	*
30	*	*	*	*	*	*	*	*	*	*	*	*	*	*	17	*	*	*	0	*	*	*	*	8	[0]	*	*	*	*
31	10	*	*	*	*	0	*	*	*	*	0	*	*	*	33	*	*	*	33	*	0	33	*	17	8	17	*	8	*
34	[0]	*	*	*	*	[0]	*	*	*	*	*	*	*	*	25	*	*	*	0	*	*	*	*	*	*	[0]	*	[0]	*
35	[0]	*	*	*	*	10	*	*	[0]	*	0	*	*	*	17	*	*	*	17	*	*	0	*	8	*	8	*	0	*
36	*	*	*	*	*	*	*	*	[0]	*	*	*	*	*	25	*	*	*	8	*	*	8	*	0	*	*	*	8	*
37	20	*	*	8	*	20	0	*	10	*	25	[0]	*	*	25	8	17	*	42	*	17	42	25	25	25	33	*	25	*
38	*	*	*	*	*	*	*	*	[0]	*	*	*	*	*	0	*	*	*	0	*	*	17	*	0	[0]	*	*	*	*
39	30	*	*	17	*	20	8	*	10	*	33	[0]	*	*	33	*	17	8	42	*	25	33	25	33	42	42	8	33	*

↑

(a) W_1

i＼j	4	5	8	15	24	32	33	40
4	*	17	[8]	*	[0]	*	*	*
5	*	*	[0]	10	[0]	*	[0]	[0]
8	*	0	*	0	*	*	0	10
15	[10]	*	[0]	*	*	*	[0]	[0]
24	0	0	8	8	*	0	8	0
32	30	10	10	8	0	*	8	8
33	20	0	[0]	0	*	*	*	0
40	30	0	*	0	[0]	*	0	*

(b) W_2 →

第５．１図　多数決行列Ａの小行列W_1、W_2

る。なお、重みは比率行列Mから、$m_{ij} \geqq 0.5$ のものに対して

$$(m_{ij} - 0.5) \times 100 \qquad (5.9)$$

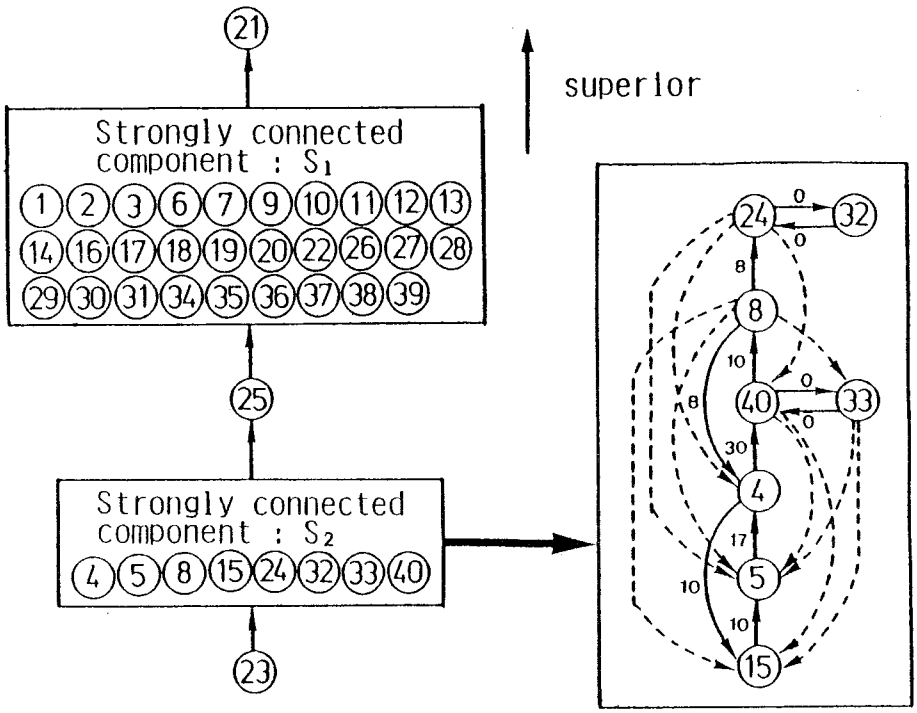
として定めた。行列W_1,W_2の節点数、枝数および虫食度は第5.1表に示す。

5.4 半順序構造化アルゴリズムの適用(7),(8)

つぎに、これら2つの強連結成分S_1、S_2に含まれる合計37名の学生に対するより詳細な順位付けを行うために、第4章で示した半順序構造化アルゴリズムを適用して、これらの強連結構造の半順序構造化を試みる。なお、この適用例の場合は長さ2のサイクル($v_i R v_j$かつ$v_j R v_i$)は同値関係と見なして抽出の対象から外し、長さ3以上のサイクルのみを抽出して半順序構造化を行った。その際、部分問題P^k の制約式の選択個数q は(4.8)式に基づいてその値を設定し、部分問題P^k の目的関数の係数w_jとしては第5.1図の行列W_1およびW_2を用いた。

第5.1表　半順序構造化アルゴリズムの適用結果

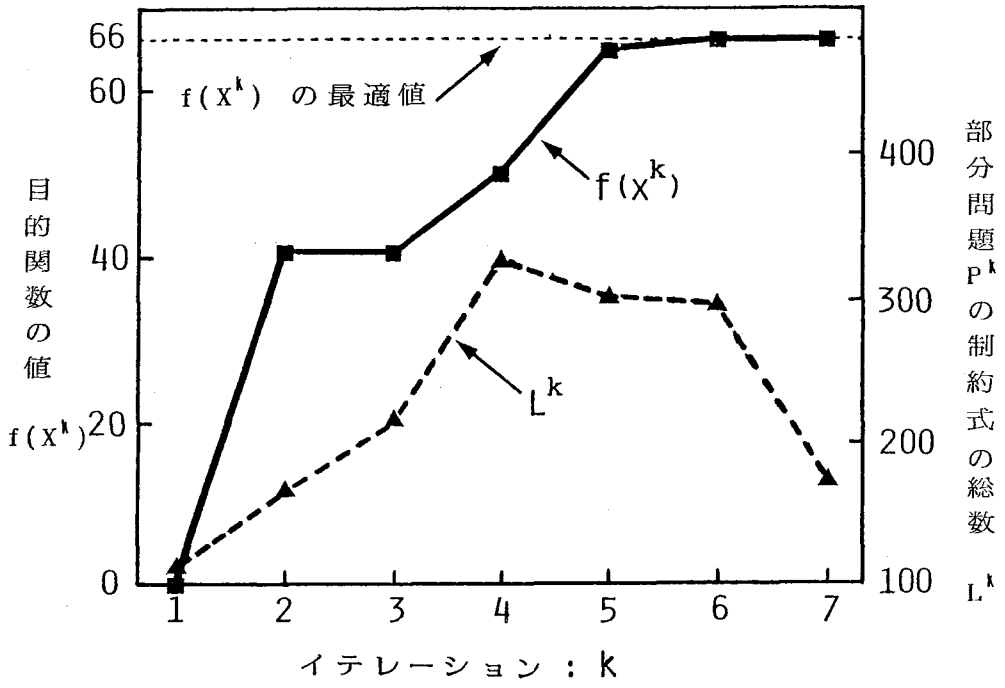
		強連結成分	
		S_1	S_2
規模	節点の数 : l	29	8
	枝の数 : n	450	40
	虫食度 : $1 - n/(l^2 - l)$	0.446	0.286
	サイクルの数 : m	$>10^7$	909
結果	制約式の選択個数 : q	113	15
	イテレーション回数	7	3
	除去された枝の数	48	12
	分割されたレベルの数	26	6
	演算時間［秒］	42.999	0.057



(a) 一対比較多数決による実験結果　　(b) S_2の最適な半順序構造化

第５．２図　口頭試問におけるランキング問題の多階層有向グラフ

その結果、２つの強連結成分はそれぞれ第5.1図の□で囲んだ枝を除去することにより半順序構造化が可能となった。結果を第5.1表、第5.2(b)図および第5.3図に示す。第5.2(b)図は強連結成分S_2の半順序構造化結果である。枝の横の数値はその枝の重みを、またフィードバック枝は除去された関係を表している。特に破線のフィードバック枝は重みが０のものである。一方、第5.3図は強連結成分S_1の半順序構造化において、アルゴリズムの各イテレーションで部分問題P^kの目的関数値および制約式集合L^kがいかに変化したかを示したものである。同図より、各イテレーションで制約式を q(＝113)個ずつ新たに追加していっても、アルゴリズムの中で不活性となった制約式集合D^kを集合L^kから逐次除外していくため、部分問題P^kの制約式数は



第５．３図　各イテレーションにおける部分問題P^kの目的関数値$f(x^k)$と制約式数L^kの変化

イテレーションを通じてさほど増加しないことが分かる。また、目的関数値は単調に最適値に収束していくことも分かる。

以上のように半順序構造化を行うことによって、第5.2(a)図のグラフがさらに35のレベルにまで分割され、40名の学生に対する７名の試問委員の集団としての選好構造を浮き彫りにすることができた。第5.2(b)図を見れば、強連結成分S_2に含まれていた８名の学生のうち、学生24と学生32あるいは学生33と学生40が実際には同順位であったことが分かる。また、学生４と学生８の関係および学生４と学生15の関係がS_2を強連結構造にしていた最大の原因であったことなども同図より明らかになる。このような除去された関係については、各委員が再度検討を加えて、話し合いにより意見を調整する必要があることをこの結果は示唆しているといえる。

５．５　結　言

半順序構造化アルゴリズムの実際問題への応用の一つとして、本章では一対比較多数決を基礎としたランキング問題を取り上げて適用を行った。ここで得られた成果を要約すると、つぎの通りである。

（１）　一対比較多数決に基づく意思決定者集団の選好構造は、「投票のパラドックス」や「個人の判断誤差」が原因して往々に強連結構造になることを、７名の試問委員の方々の御協力を得て、実際に口頭試問における学生間の順位付け実験を行うことにより実証した。

（２）　つぎに、この実験で得られた７名の試問委員の集団選好構造に半順序構造化アルゴリズムを適用して、その中に含まれていた二つの強連結成分の半順序構造化を行った。その結果、一対比較多数決だけでは十分な順序付けのできなかった非推移的な学生間の順位関係もさらに詳細に順位付けすることができることを明らかにした。そうして、本アルゴリズムが集団意思決定過程における合意形成のための支援手法として有効であることを示した。

第５章の参考文献

（１）　V.J.Bowman and C.S.Colanton : Majority Rule Under Transitivity Constraints, Management Science, Vol.19, No.9, pp.1029-1041 (1973)

（２）　K.J.Arrow : Social Choice and Individual Values, 2nd ed.; Yale University Press, New Haven (1963)

（３）　守安,井上 : 推移律を満足する社会選好の合成法，システムと制御，Vol.26, No.8, pp.509-515 (昭 57)

（４）　増田,藤井 : 緩和法の概念に基づく強連結システムの半順序化とその応用，計測自動制御学会 第2回知識工学シンポジウム論文集，pp.45-50 (昭 59)

（５）　増田,新山,藤井 : 強連結システムの半順序構造化とその応用，シス

テムと制御，Vol.28，No.8，pp.544-550（昭 59）

（6） 井上,守安,木村 ： 推移律を満足する集団選好の構造化，計測自動制御学会論文集，Vol.18，No.9，pp.905-911（昭 57）

（7） 増田,新山,平峰,藤井 ： 強連結構造の半順序構造化アルゴリズムのランキング問題への応用，日本自動制御協会 第28回システムと制御研究発表講演会講演論文集，S3（昭 59）

（8） 増田,新山,藤井 ： 緩和法の概念に基づく強連結システムの半順序構造化，計測自動制御学会論文集，Vol.20，No.9，pp.814-821（昭 59）

第6章　シミュレーションにおける計算手順決定問題への応用

6.1　緒　言

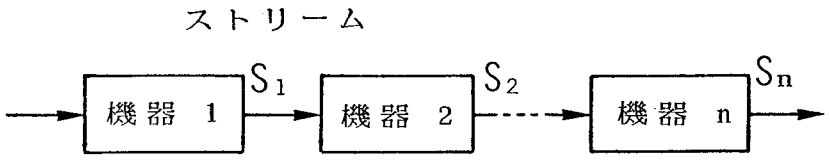
半順序構造化アルゴリズムの実際問題へのもう一つの応用例として、本節ではプラントシミュレーション(plant simulation)における計算手順決定問題(decision problem of calculation process)への適用を行う。ここでいう計算手順決定問題とは、数学モデルを用いたプラントシミュレーションの際の数値計算(繰り返し計算)において、最も収束効率の良い繰り返し計算の手順を求める問題のことを意味する。

以下、本章では、まず6.2節でプラントシミュレーションにおける計算手順決定問題について概説し、大規模強連結構造をもつプラントシステムのシミュレーション計算においては、なぜモデル構造を半順序構造化しなければならないのか、その必要性を明確にする。つぎに、6.3節では実際の硫酸プラントモデルを例にとり、半順序構造化アルゴリズムをそのシミュレーションにおける計算手順決定に適用した例を示す。そうして、この種の問題にも本アルゴリズムが有効で、かつ実用的であることを明らかにする。

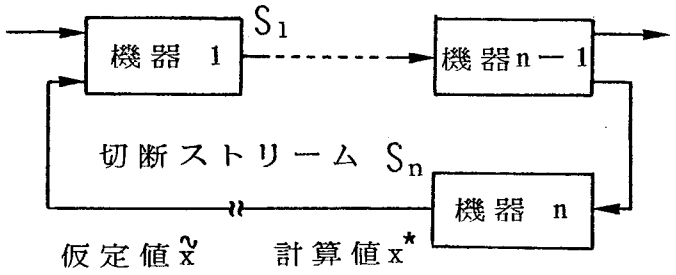
6.2　計算手順決定問題

プラントシステムの管理、生産性の向上化を図る目的で、通常種々のシステムズアプローチ[(1)]を用いたプラントシステムのモデル化が行なわれる。このモデル化においてはプラントを構成する諸設備あるいは機器間の物質や熱の流れ(ストリーム(stream))を入出力変数として表し、これらの変数を用いてプラントシステムにおけるさまざまな収支関係を数学モデルの形に構築する。

このようにして作成されたプラントモデルの構造が第6.1(a)図のようにサイクルの存在しない場合であると、そのシミュレーションにおける計算過程は入力側から設備あるいは機器に対応するサブモデルごとに数値計算を順次進めていくことで、プラント全体のシミュレーション結果(数値解)を得るこ



(a) サイクルのない場合



(b) サイクルのある場合

第6．1図 プラントシステムの構造

とができる。ところが第6.1(b)図のようにサイクルの存在する場合であると、そのサイクルに含まれるサブモデル群は一度に数値計算を行わなければならない。そのため、モデルが大規模である場合や非線形性を有する場合には式および変数の数が膨大な非線形連立方程式を一度に解かなければならず、その結果数値解を得るのに多大な計算量を要したり、場合によっては求解自体が困難になることがある。このような場合には、モデル全体を一度に解く代わりにサイクルを構成するストリームの一部を切断して(例えば、図中のS_n)このストリームに含まれる変数に仮定値$\tilde{x}$ を与えて計算を進め、一巡して得られる計算値x^*が$\tilde{x}$ に収束するまで仮定値を変更するといった繰り返し計算(iterative calculation)が通常採られる。[2)～(4)] このような繰り返し計算を実行するためには、モデルの構造を半順序構造化して、しかもできるだけ計算効率が良くなるように(換言すれば、繰り返し回数を少なくするように)その計算手順を決定する必要がある。

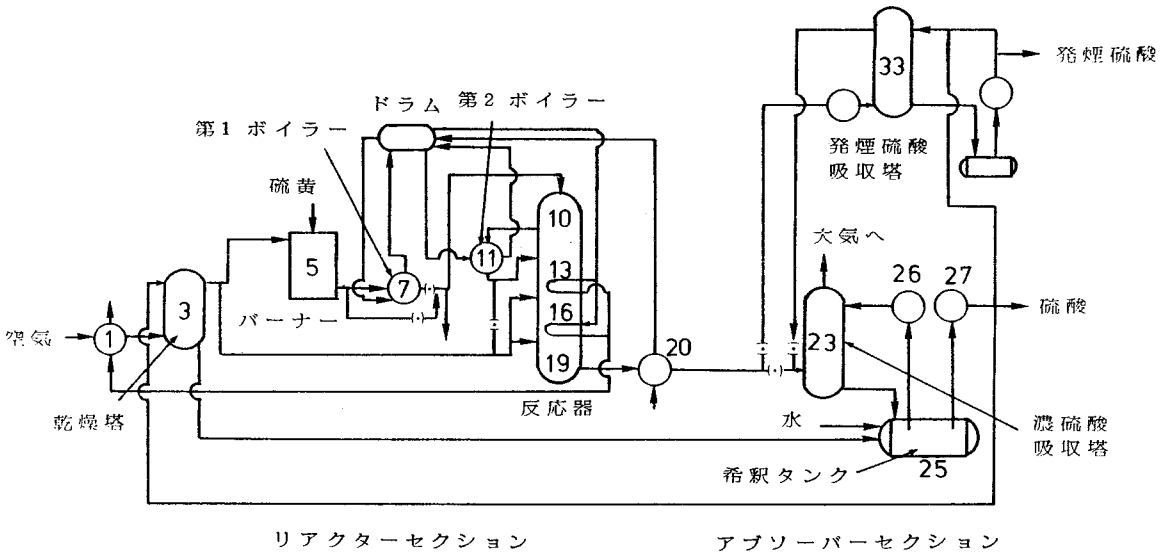
そこで次節ではShannonらによって作成された接触式硫酸プラントモデル(contact sulfuric acid plant model)[(5)] を例にとり、そのシミュレーショ

ンの際の最適な計算手順を決定するために半順序構造化アルゴリズムを適用する。

6．3　硫酸プラントモデルへの適用

6．3．1　接触式硫酸プラントモデル(5)

Shannonらが作成したモデルは、Canadian Industries Ltd. の接触式硫酸プラントを対象としたものである。このプラントは二つの大きなセクションから構成されている。一方は二酸化硫黄SO_2を五酸化バナジウムV_2O_5の触媒の下で酸化させて三酸化硫黄SO_3とするリアクターセクション(reactor section)で、他方はそのSO_3を吸収するアブソバーセクション(absorber section)である。リアクターセクションはさらに送風機、乾燥塔、硫黄バーナー、転化器、蒸気発生システムなどから、またアブソバーセクションは空気乾燥塔、濃硫酸吸収塔などから構成される。第6.2図は硫酸プラントのプロセスフロー図で、モデル化にあたって考慮した主要設備を示している。本プラントは発生したエネルギーなどを有効に再利用しており、第6.1(b)図



第６．２図　硫酸プラントのプロセスフロー図

に示したようなフィードバック構造を持っている。このため、全体の数学モデルはおよそ1000の変数および200のパラメータを含んだ約500の非線形連立方程式から構成されている。第6.3図はその情報の流れを有向グラフで表現したものである。節点はプラントの諸設備(ただし、これは蒸留、抽出、反応、吸収といった最小単位の化学変化が起こっている生産設備を指す。例えば、希釈タンクは節点24の混合器と節点25のポンプで構成される)に対応する数学モデルを表し、枝はこれらの設備に出入りする情報ストリーム(入出力変数)を表している。ただし、同図中のシミュレーション実行セクション(executive section)はシミュレーションを行うときにパラメータの自動変更などを行うセクションで、節点41から節点44に対応する現実の設備は存在しない。

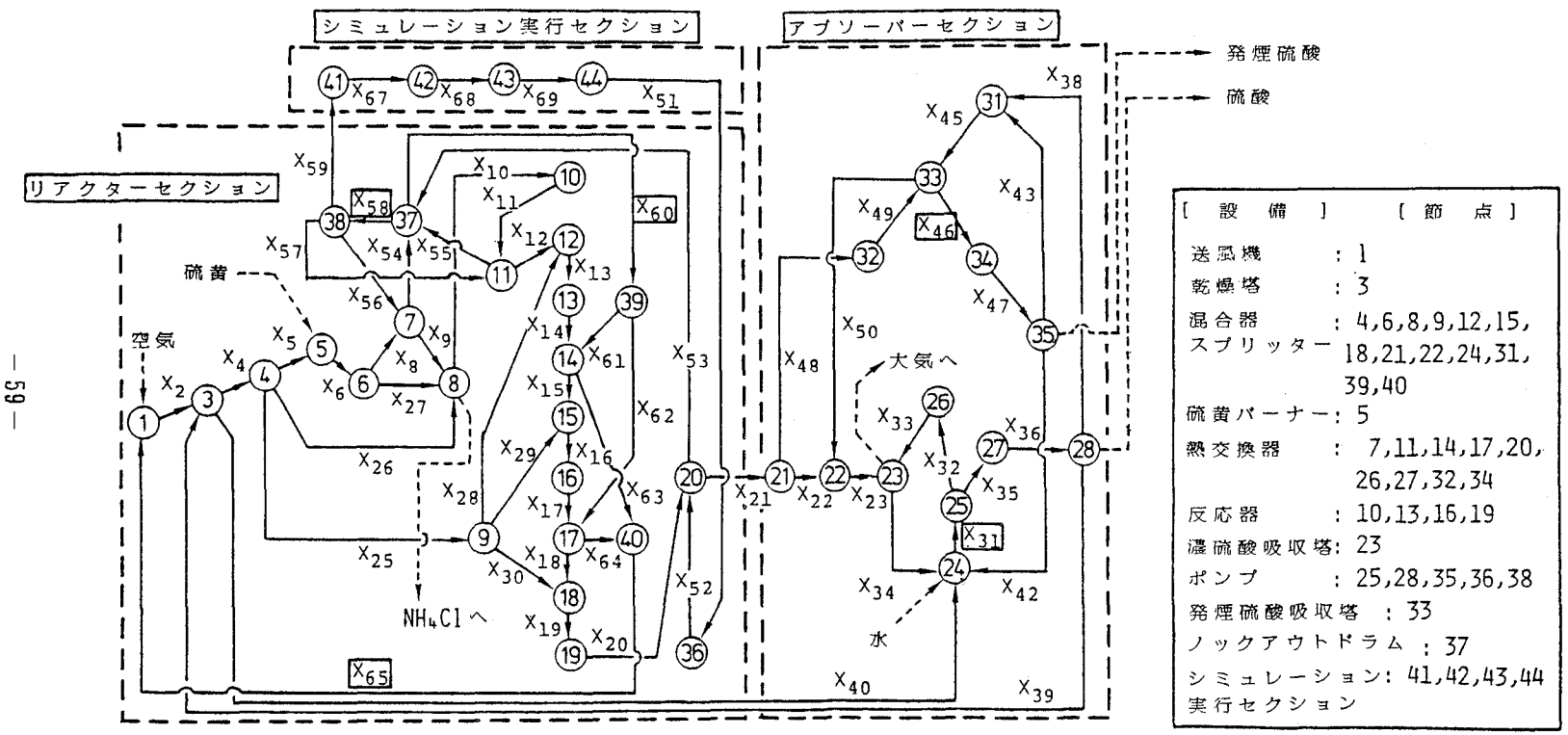
このグラフは節点数$l=41$、枝数$n=61$、虫食度$1-n/(l^2-1)=0.963$で、グラフ全体が強連結になっている。そして、このグラフには第6.1表に示す長さ3から26の合計97個のサイクルが含まれている。この硫酸プラントのシミュレーションを繰り返し計算を採用しないで実行しようとすると、非常に非線形性の強い約500の非線形連立方程式を一度に解かなければならず、いくら高性能の大型計算機を駆使して計算したとしても膨大な演算時間を必要とする。したがって、本硫酸プラントモデルにおいてはどうしても繰り返し計算を用いてシミュレーションを行わなければならず、このため繰り返し計算の手順を決定する必要が生じる。

第6．1表　硫酸プラントモデルに含まれるサイクルの分類

サイクルの長さ ： k	3～5	6～10	11～15	16～20	21～25	26
サイクル数	5	13	22	16	35	6

6．3．2　適用結果[6),(7)]

最適計算手順決定のために、このグラフに半順序構造化アルゴリズムを適用して構造の半順序構造化を試みる。なお、適用に際して部分問題P^kに逐



第６．３図　硫酸プラントモデルの有向グラフ

次追加する制約式の選択個数q の値は(4.8)式に基づいてq＝20と設定した。また目的関数は、最少数のストリーム除去による半順序構造化問題と考えて、その係数の値は総て1とした。その理由は、除去されたストリームに含まれる変数、すなわち仮定値を与える変数の個数が少ないほど繰り返し計算の収束が速くなるという考えによる。

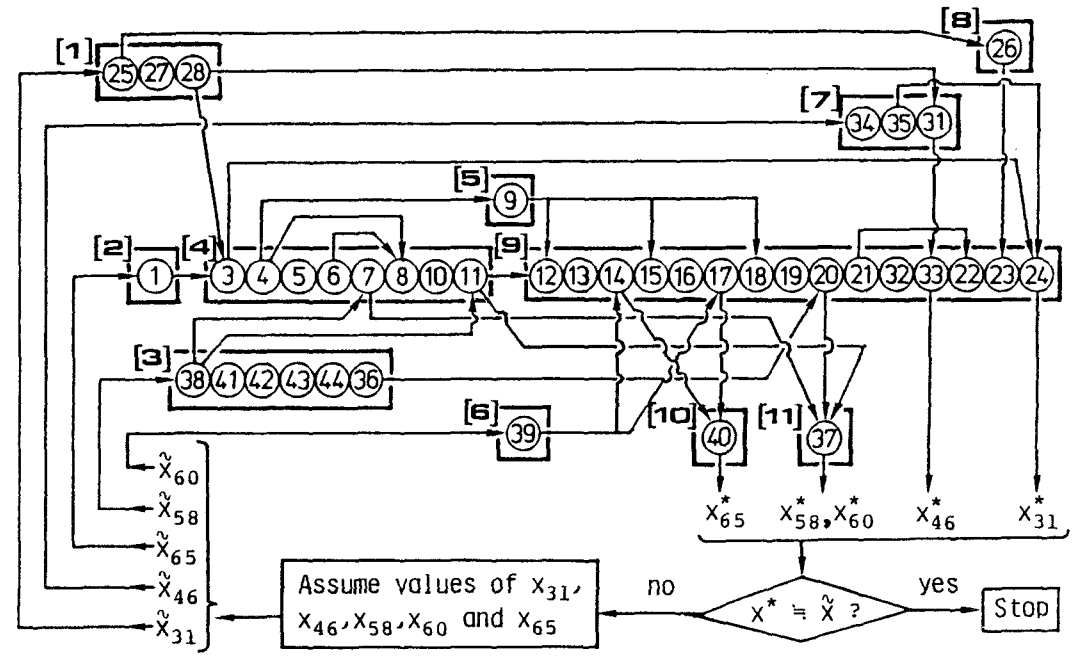
その結果、本適用例の場合、最適な半順序構造を与える除去枝集合として、つぎの二組の集合が求められた。

最適除去枝集合Ｘ1 ： $\{ X_{31}, X_{46}, X_{58}, X_{60}, X_{65} \}$ （6．1）

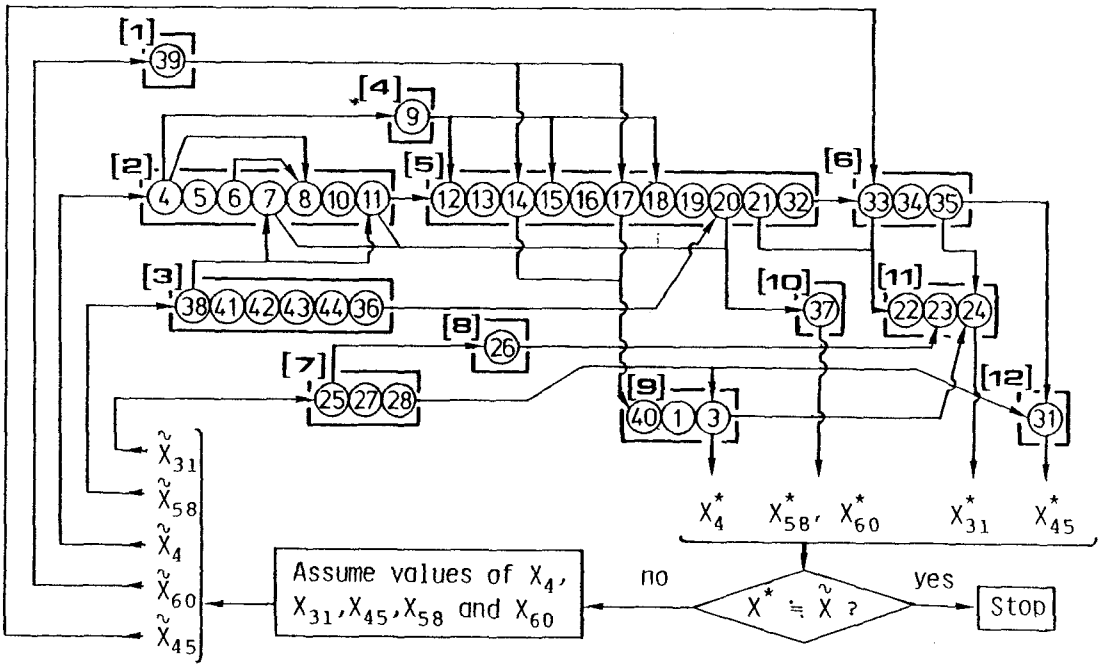
最適除去枝集合Ｘ2 ： $\{ X_{4}, X_{31}, X_{45}, X_{58}, X_{60} \}$ （6．2）

第6.3図の□で囲んだストリームは最適除去枝集合Ｘ1に対応するものである。なお、計算に要したイテレーション回数は上記いずれの場合も1回で、演算時間はそれぞれ0.170秒、0.217秒であった。ここでイテレーション回数が1回で済んだのは、このグラフの場合、虫食度が0.963と大きく、そのためグラフに含まれるサイクル数、すなわち問題Ｐの制約式数が97と少なかったことに起因しているものと考えられる。

これらの結果に基づく最適な繰り返し計算手順を第6.4図に示す。図中の鍵かっこ内の数字はシミュレーションの際の計算手順を示している。すなわち、この番号の若いブロックに属する規模の小さいサブモデルを、左側から順次カスケード的に数値計算を行い、これらの手順を繰り返すことによって、最終的にモデル全体のシミュレーション結果を得ることができる。第6.4(a)および(b)図に示すいずれの計算手順においても、仮定値を与える変数は5つのストリームに属する5個の変数だけでよく、その結果、数回の繰り返しで計算は収束するものと期待できる。また、一回のイテレーションにおいて一度に解かなければならない最大の非線形連立方程式は、節点10の反応器を表すサブモデルの式数22個の連立方程式である。これは繰り返し計算を用いずに式数約500の非線形連立方程式を一度に解くことに比べると、格段に容易であり、しかも短時間で計算をすることができる。



(a) 最適除去枝集合X1に基づく場合



(b) 最適除去枝集合X2に基づく場合

第6．4図　シミュレーションにおける最適な繰り返し計算手順

以上に述べたように、半順序構造化アルゴリズムを用いてモデル構造を半順序化して最適な繰り返し計算の手順を求めることにより、大規模な硫酸プラントのシミュレーションも比較的簡単に行うことが可能になった。なお、本適用例では目的関数の重み係数は総て等しく設定して最適計算手順を求めたが、対象とするモデルによっては最適除去枝として選ばれたストリームに関して物理的知識が不足していて、適切な仮定値を与えられない場合がある。こういった場合には、仮定値を設定しやすいストリームに小さい重みを、逆に設定しにくいストリームには大きい重みを付加して目的関数を作成し、本アルゴリズムを適用することが考えられる。

6.4 結　言

本章では、プラントシミュレーションにおける計算手順決定問題を取り上げて半順序構造化アルゴリズムの適用を行った。ここで得られた結果を要約すると、つぎの通りである。

（1） まず最初に、大規模強連結構造をもつシステムのシミュレーション計算においては、一度に数値計算を行うことは規模の大きさあるいは非線形性が原因して極めて困難であり、通常、繰り返し計算によってシミュレーションを行わなければならないことを具体的に示した。そうして、この繰り返し計算を実行するためには、システムの一部のストリームを除去し構造を半順序構造化して、その計算手順を決定する必要があることを明確にした。

（2） つぎに、実際の接触式硫酸プラントモデルを取り上げて、半順序構造化アルゴリズムによりそのシミュレーション計算手順の決定を試みた。その結果、本適用例の場合、二つの最適計算手順が存在することが明らかになった。さらに、これらの手順に従ってシミュレーションを行えば、小規模な数値計算を繰り返すだけで容易にシミュレーション結果を得ることができることを示した。

第 6 章の参考文献

（1） 例えば、計測自動制御学会編 ： 自動制御ハンドブック 基礎編，オーム社（昭 58）

（2） D.V.Steward : Partitioning and Tearing of Large Systems of Equations, SIAM J. Numer. Anal., Vol.2, No.2, pp.345-365 (1965)

（3） 寺野,グエン ： ボトルネック法による強連結グラフの分割法とその応用，計測自動制御学会論文集，Vol.12, No.6, pp.681-686（昭 51）

（4） 小沢,新谷 ： リサイクルシステムにおけるループ切断の新方法―図心逐次拡大法―，電気学会論文誌，Vol.103-C, No.5, pp.109-116（昭 58）

（5） P.T.Shannon et al. : Computer Simulation of a Sulfuric Acid Plant, Chemical Engineering Progress, Vol.62, No.6, pp.49-59 (1966)

（6） 増田,新山,藤井 ： 強連結構造の半順序化アルゴリズムの計算手順決定問題への応用，日本自動制御協会 第28回システムと制御研究発表講演会講演論文集，S4（昭 59）

（7） 増田,新山,藤井 ： 緩和法の概念に基づく強連結システムの半順序構造化，計測自動制御学会論文集，Vol.20, No.9, pp.814-821（昭 59）

第7章　ビルディングブロック方式ＬＳＩの配線問題への応用

７．１　緒　言

大規模集積回路ＬＳＩ(large scale integration)の集積度は毎年約２倍の速度で確実に増加していると言われている。このようなＬＳＩの高集積化、高性能化に対処するためには、ＬＳＩのチップ面積をできるだけ小さくするような、計算機援用による高度のレイアウト設計技術ＣＡＤ(computer aided design)が必要不可欠である。そこで本章では、半順序構造化アルゴリズムの実際問題へのもう一つの応用例として、ビルディングブロック方式ＬＳＩの配線問題を取り上げて、そのレイアウト設計におけるチップ面積最小化に半順序構造化アルゴリズムを適用することを考える。

以下、本章ではまず7.2節で対象とするビルディングブロック方式ＬＳＩおよびその代表的な配線方法である幹線支線方式について説明する。続いて7.3節では、この幹線支線方式による配線設計においては幹線の上下関係の制約を表す制約グラフに含まれるサイクルを解消することが重要な問題になり、この問題解決に3.4.2で述べた節点除去による半順序構造化が必要となることを示す。つぎに7.4節では、実際の配線要求に対して半順序構造化アルゴリズムを用いて配線設計を行った例を示し、アルゴリズムの有効性を明らかにする。

７．２　ビルディングブロック方式ＬＳＩの配線設計

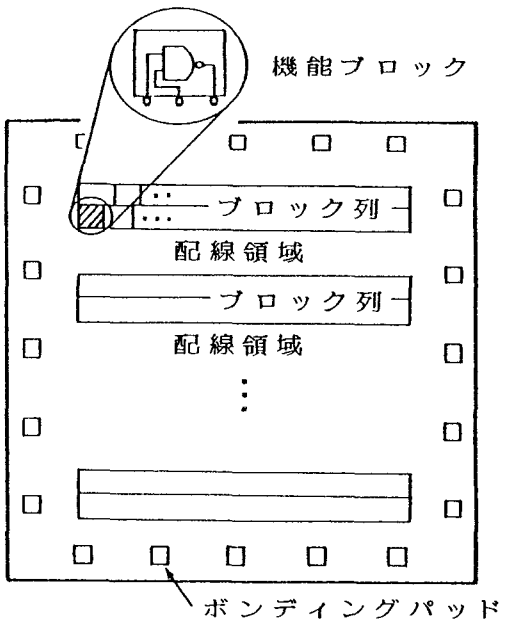
所定の機能をもつＬＳＩを短時間に誤りなく実現するためのレイアウト方法(design approach for layout)としてビルディングブロック方式(building block approach)、[1] 一次元アレイ方式(one dimensional array approach)[2]、ＰＬＡ方式(programable logic array approach)、[3] マスタスライス方式(masterslice approach)[4] などが提案されている。これらの中でもビルディングブロック方式はＭＯＳＬＳＩでは最も広く用いられている方式で

ある。本節では、ビルディングブロック方式の概略とこのレイアウト方法を採用したときの代表的な配線方法である幹線支線方式(trunk and branch method)[5] について述べる。

7．2．1　レイアウト方法：ビルディングブロック方式[1]

ＬＳＩのレイアウト方法の一つにビルディングブロック方式がある。このビルディングブロック方式とは、数個のゲート(gate)を組み合わせた程度の機能を持つ機能ブロック(function block)の回路パターンがあらかじめ多数設計されてライブラリとして登録されており、いったんＬＳＩで実現すべき論理が与えられるとこれらの機能ブロック単位でチップ(chip)上の配置を決定し、機能ブロック間の相互配線を行う方式のことである。

ビルディングブロック方式ＬＳＩの概略を第7.1図に示す。チップは同図のように周辺のボンディングパッド領域(bonding pad region)、機能ブロックが規則正しく並べられた横に細長いブロック列領域(block array region)およびそれらの間の配線領域(interconnection region)に区分される。一個の機能ブロックの幅は自由であるが、高さは一定に揃えられブロック列内の相互入れ換えを容易にしている。また、各機能ブロックの外部(他の機能ブロックあるいはボンディングパッド)への配線端子はブロック列の上辺または下辺に限定されている。隣接ブロック列間の配線は、それら



第７．１図　ビルディングブロックＬＳＩの概略図

ブロック列にはさまれた配線領域で行なわれるが、それ以外のブロック列間の配線は、途中のブロック列を通過するか(貫通ブロック[6])、外周を迂回することにより行なわれる。[7]

このビルディングブロック方式によるレイアウト設計の目的は、できる限りチップ面積を小さくすることである。ところがブロック列領域、ボンディングパッド領域の大きさは決まっているので、チップ面積を最小にすることは配線領域面積を最小にすることで実現できる。しかも配線領域のブロック列に平行な方向の長さはブロック列の長さで規定されてしまうため、結局、チップ面積の最小化はブロック列に垂直な方向の長さ、すなわち配線領域幅を最小にすることに帰着される。配線領域幅を最小化するに当たっては、

i) 各機能ブロックをどのブロック列のどの場所に配置するか、

ii) ブロック列間の配線をどのように行うか

を総合的に考慮して行う必要がある。[8]

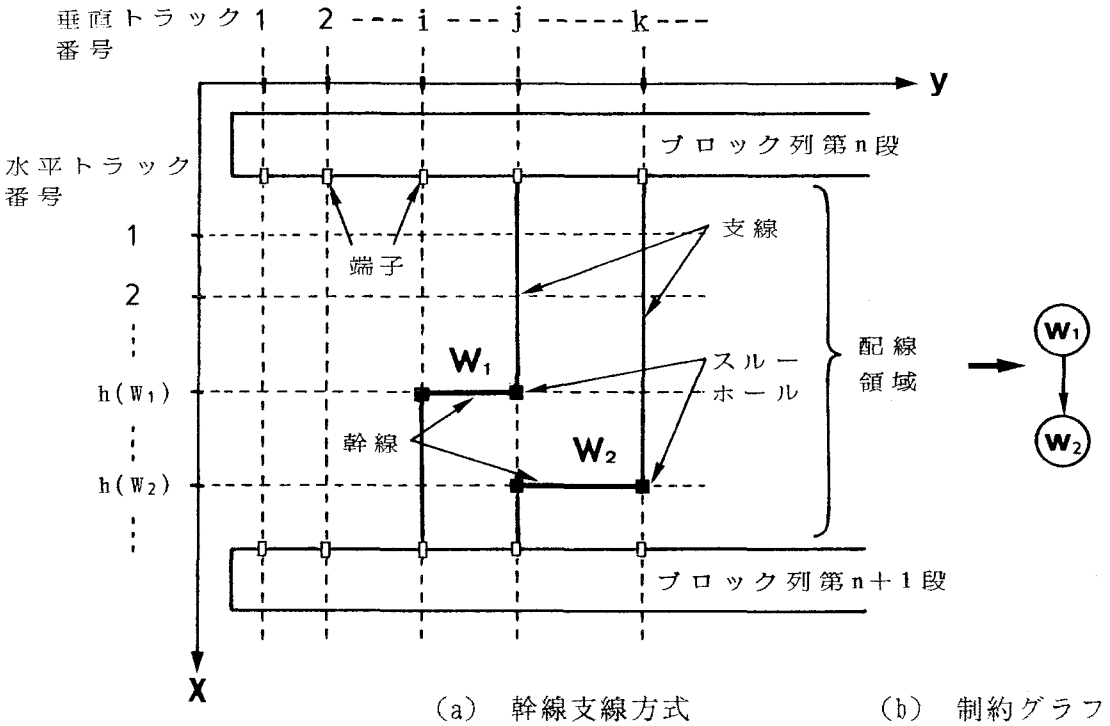
7．2．2 配線方法 ： 幹線支線方式[5]

ここでは、前項のi) の問題が既に解決されている、すなわち各機能ブロックの位置が既に定まっており、隣接するブロック列の端子の接続情報(以後、配線要求と呼ぶ)が与えられているものとして、この配線要求を実現するための配線方法(幹線支線方式)について述べる。

幹線支線方式(trunk and branch method)とは、配線経路をブロック列に平行な方向(y方向)の1本の線分と、垂直な方向(x方向)のいくつかの線分とで構成する方式である。

第7.2(a)図に幹線支線方式による配線の一例を示す。接続すべき端子はブロック列に沿って向き合って並んでいる。また、配線領域には整数値単位の格子目状に仮想的な配線トラックが設けられており、y方向のものを水平トラック(horizontal track)、x方向のものを垂直トラック(vertical track)と呼ぶ。水平トラックの必要な線分(これを幹線(trunk)と呼ぶ)にはアルミニウム膜を蒸着し、垂直トラックの必要な線分(これを支線(branch)と呼ぶ)

には低抵抗を拡散して配線を行う。すなわち、幹線、支線は二層配線(two layer interconnection)されていることになり、接続する必要があるときにはスルーホール(through hole ; 図中の■印)を用いて接続される。



(a) 幹線支線方式　　(b) 制約グラフ

第７．２図　幹線支線方式と制約グラフ

７．３　配線問題における制約グラフ

前節で述べた幹線支線方式を用いれば、幹線の位置(どの水平トラック上に配置するか)が定まったとすると、そこから接続すべき端子に垂線を下ろしてそれを支線とすれば配線は一意的に決定する。すなわち、配線経路を幹線の位置で代表することが可能となり、配線問題は幹線の位置割り当て問題に帰着させることができる。

このとき、幹線同志あるいは支線同志は重なってはいけないという要請から、幹線の位置割り当てに制約が生じる。すなわち、第7.2図の例では、配

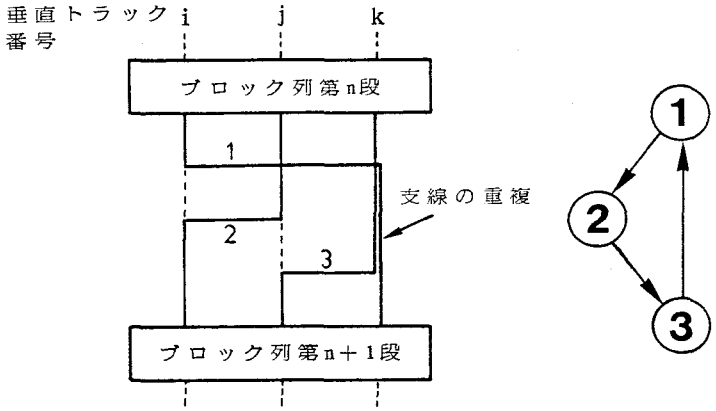
線W_1、W_2の使用する水平トラックの間には、W_1のトラックはW_2のトラックより上に置くという関係を保つ必要がある(W_iの使用する水平トラックの番号を$h(W_i)$で表すと$h(W_1)<h(W_2)$となる)。これはW_1とW_2の使用する垂直トラックが互いに番号j のものに一致することを防止するためである。このような水平トラック使用における上下関係は制約グラフで表現することができる。すなわち、各々の幹線を節点で表し、水平トラック使用に関して$h(W_i)<h(W_j)$という関係があれば、これを節点W_iから節点W_jに向かう有向枝で表せばよい。第7.2(b)図は同(a)図に対応する制約グラフを示している。

この制約グラフをもとに、番号の若い水平トラックに順次重ならないように幹線をつめて配置していけば、各幹線の位置を決定することができる。[9]

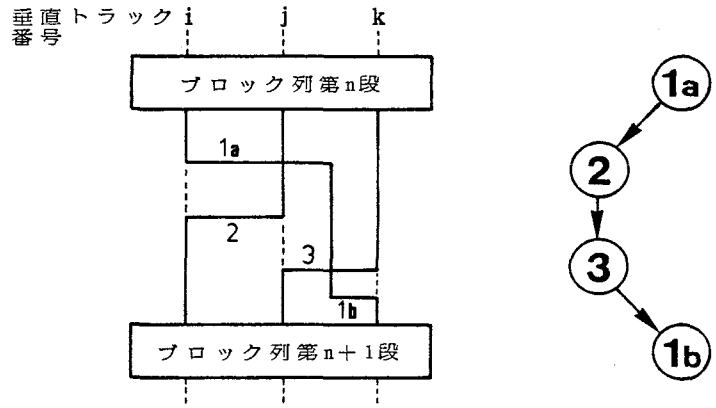
ところが配線要求によっては、このような配線が不可能になる場合が発生する。その一例を第7.3(a)図に示す。同図において、配線経路1と配線経路2は同じ垂直トラックi を使用しているので、幹線2は幹線1より下に置かなければならない。同様に幹線3は幹線2より下に置かなければならないが、その結果、垂直トラックk において配線経路1と3の支線が重複してしまう。これは幹線1、2、3の上下関係において推移律が満たされていないことに起因しており、それは制約グラフにおいてはサイクルが発生していることに対応する。言い換えれば、制約グラフにサイクルが存在すると、それはどのように幹線を配置してもどこかで支線が重複し、配線できないことを意味する。

配線を可能とするためには、第7.3(b)図のようにいづれかの幹線(例えば、同図の幹線1)を分割して、幹線の上下関係における非推移性を除去する必要がある。これは制約グラフにおいて、節点を分割して(例えば、1aと1b)、サイクルを解消することに相当する。すなわち、節点を分割して、その節点に接続する枝を無視することによりサイクルを解消する、いわゆる3.4.2で述べた節点除去による強連結構造の半順序構造化を意味する。ところで幹線の分割(グラフで言えば、節点の除去および分割)という処置は、必要な水平トラック数の増加につながるので、「配線領域幅を最小にする」という観点か

らはできるだけ少ないほうが望ましい。したがって、幹線支線方式による配線問題においては、最少数の節点除去による制約グラフの半順序構造化を行う必要がある。



(a)　分割前(配線不可能)



(b)　幹線1の分割後(配線可能)

第7．3図　幹線分割による配線実現化

7．4　半順序構造化アルゴリズムの適用[10),(11)]

7．4．1　問題の設定

7.2.1で述べたように、配線領域幅の最小化を図るためには機能ブロック

の配置とその相互配線の両方を考慮しなければならないが、以下で述べる適用例では、既に機能ブロックの配置が定まっており、どの端子間の接続を行うかという配線要求が与えられているものと仮定する。

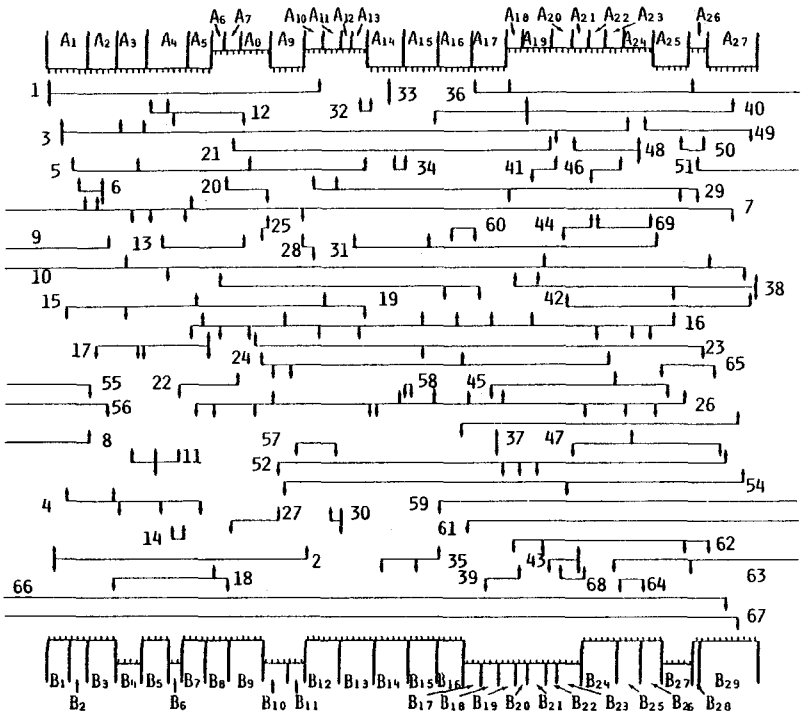
ここで対象とする配線問題は、文献(12)で取り扱われている実際のビルディングブロック方式ＬＳＩチップ内の、一対の隣接ブロック列間の配線問題(例題１)および筆者が独自に作成した配線問題(例題２)の二つである。各々の例題における配線要求を第7.1表および第7.4図に示す。第7.1表は例題１の配線において接続すべき端子の位置を示したものである。Ａ、Ｂはそれぞれ上、下段のブロック列内にある機能ブロックを表し、その右上、右下の添字はそれぞれ左側からつけた各機能ブロックの端子番号および機能ブロック番号を表している。例えば、A_5^2は上段のブロック列内のA_5なる機能ブロッ

第７．１表　例題１の配線問題における配線要求

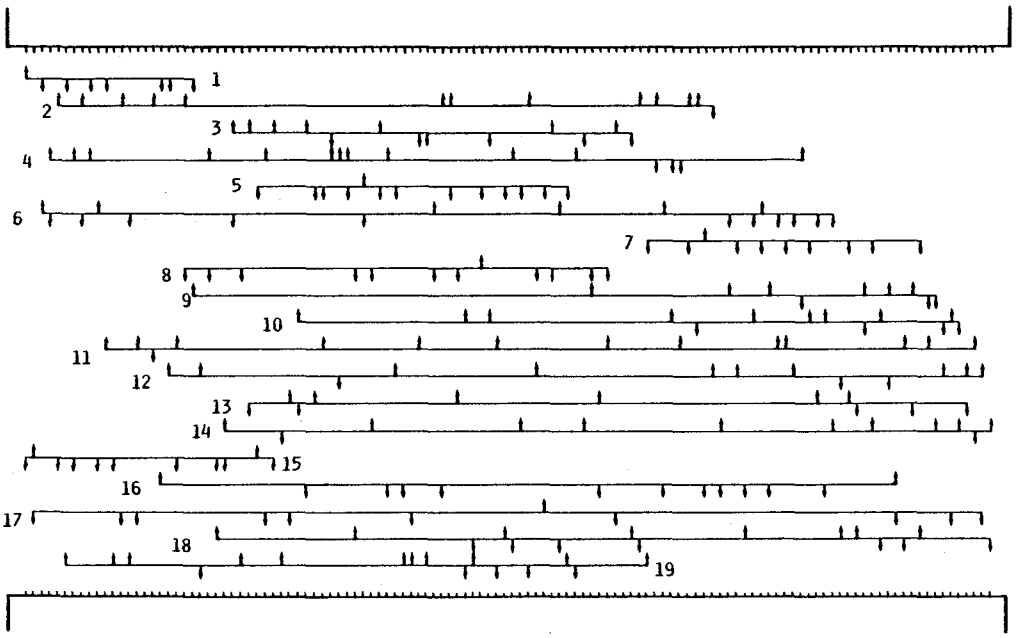
配線名	接続端子位置	関連サイクル	配線名	接続端子位置	関連サイクル
1	A_1^1, A_{10}^3, B_1^1		12	$A_4^1, A_4^4, B_6^1, B_9^3$	
2	A_1^2, A_{10}^1, B_1^2		13	A_4^3, A_8^1	
3	$A_1^3, A_3^1, A_3^5, A_{24}^1$	a,b	14	A_4^5, A_4^7	
	B_1^3, B_{22}^2		15	$A_5^2, A_{11}^1, B_1^4, B_4^2$	a,c,e
4	$A_1^4, A_2^5, B_4^1, B_5^4$	a,c		B_{13}^5	
	B_7^4		16	$A_5^3, A_9^3, A_{15}^4, A_{16}^4$	a,b,c
5	$A_1^5, A_3^4, A_8^2, A_{13}^3$			$A_{17}^4, A_{19}^2, A_{25}^4, B_7^2$	d,e
6	A_1^6, A_2^3, B_3^3			$B_8^3, B_9^4, B_{12}^3, B_{13}^4$	
7L	$A_1^7, A_2^2, A_5^1, B_4^3$	a,b,c		$B_{24}^3, B_{25}^3, B_{26}^2$	
	$B_5^2, B_7^1, B_{11}^3, B_{29}^6$	d,e	17	$A_5^4, B_3^2, B_4^4, B_5^1$	
8L	A_2^1			B_8^1	
9L	A_2^4		18	A_6^1, B_3^5, B_8^4	f
10L	$A_3^2, A_{19}^4, A_{27}^1, B_5^5$	e	19	$A_8^2, B_{16}^2, B_{17}^3$	
	B_{29}^8		20	A_7^1, B_{10}^1	f
11	$A_3^3, A_4^2, A_4^6, B_5^3$		21	A_7^2, A_{19}^5	

第７．１表　例題１の配線問題における配線要求(続き)

配線名	接続端子位置	関連サイクル	配線名	接続端子位置	関連サイクル
22	$A_7{}^3, B_6{}^2$		43	$A_{21}{}^2, B_{23}{}^4, B_{22}{}^1$	
23	$A_8{}^3, B_{15}{}^3, B_{29}{}^1$	d	44	$A_{22}{}^1, B_{23}{}^1$	
24	$A_8{}^4, A_{16}{}^5, A_{23}{}^1, B_{10}{}^2$	f	45	$A_{23}{}^2, B_{18}{}^2, B_{27}{}^1$	
	$B_{11}{}^1$		46	$A_{23}{}^3, B_{24}{}^2$	
25	$A_8{}^5, B_9{}^6$	f	47	$A_{24}{}^2, B_{23}{}^3, B_{29}{}^4$	
26	$A_9{}^1, A_{14}{}^6, A_{15}{}^6, A_{16}{}^6$	a,b,c	48	$A_{21}{}^1, A_{24}{}^3, B_{25}{}^4$	
	$A_{17}{}^6, A_{25}{}^6, B_7{}^3, B_8{}^2$	d,e,f	49	$A_{24}{}^4, B_{29}{}^9$	
	$B_9{}^5, B_{13}{}^6, B_{14}{}^1, B_{24}{}^1$		50	$A_{25}{}^5, A_{26}{}^3$	
	$B_{25}{}^2, B_{26}{}^3$		51R	$A_{26}{}^2$	
27	$A_9{}^2, B_9{}^1$		52	$A_{27}{}^4, B_{10}{}^3, B_{19}{}^1, B_{20}{}^1$	
28	$A_9{}^6, B_{12}{}^2$			$B_{21}{}^2$	
29	$A_{10}{}^2, A_{11}{}^3, B_{19}{}^2, B_{27}{}^3$		53	$A_{27}{}^6, B_{16}{}^5$	
	$B_{28}{}^1$		54	$A_{27}{}^7, B_{10}{}^4, B_{23}{}^2$	e
30	$A_{12}{}^1, A_{11}{}^2, B_{13}{}^1$		55L	$B_3{}^1$	
31	$A_{13}{}^1, A_{15}{}^5, A_{25}{}^1$		56L	$B_3{}^4$	
32	$A_{13}{}^2, A_{14}{}^1$		57	$B_{11}{}^2, B_{12}{}^6$	
33	$A_{14}{}^4, B_{14}{}^3$		58	$B_{14}{}^6, B_{15}{}^1$	
34	$A_{14}{}^5, A_{15}{}^1$		59R	$B_{16}{}^1$	
35	$A_{16}{}^1, B_{14}{}^2, B_{15}{}^2$		60	$B_{16}{}^3, B_{17}{}^2$	
36R	$A_{17}{}^1, A_{18}{}^1, A_{26}{}^1$		61R	$B_{17}{}^1$	
37	$A_{17}{}^5, B_{18}{}^3$		62	$B_{19}{}^3, B_{21}{}^3, B_{27}{}^4, B_{29}{}^2$	
38	$A_{18}{}^2, A_{19}{}^3, A_{27}{}^9, B_{27}{}^2$		63R	$B_{24}{}^6, B_{27}{}^5$	
	$B_{29}{}^{10}$		64	$B_{25}{}^1, B_{26}{}^1$	
39	$A_{18}{}^3, B_{18}{}^1$		65	$B_{26}{}^4, B_{29}{}^3$	
40	$A_{19}{}^1, A_{27}{}^5, B_{15}{}^5, B_{20}{}^2$	a,b,c	66L	$B_{29}{}^5$	
		d,e	67L	$B_{29}{}^7$	
41	$A_{20}{}^1, B_{21}{}^1$	a,b	68	$A_{20}{}^2, A_{21}{}^3$	
42	$A_{20}{}^3, A_{27}{}^8$		69	$A_{22}{}^2, A_{24}{}^5$	



(a)　例題１のネットリスト



(b)　例題２のネットリスト

第７．４図　例題１および２の配線要求を表現したネットリスト

クの左から二番目の端子を表している。また、配線名の横の記号LおよびRはその配線が本配線領域と隣接するそれぞれ左および右側の外部配線領域とも接続していることを意味する。一方、第7.4図は各々の例題における配線要求を幹線で代表させてネットリストの形で描いたものである。各幹線の横の数字は配線名を表す。また、幹線と接続すべき端子とを結ぶ支線を矢印で示す。同図において各幹線が使用している水平トラックは仮のものであり、このままの幹線位置で100%配線を行えるものではない。

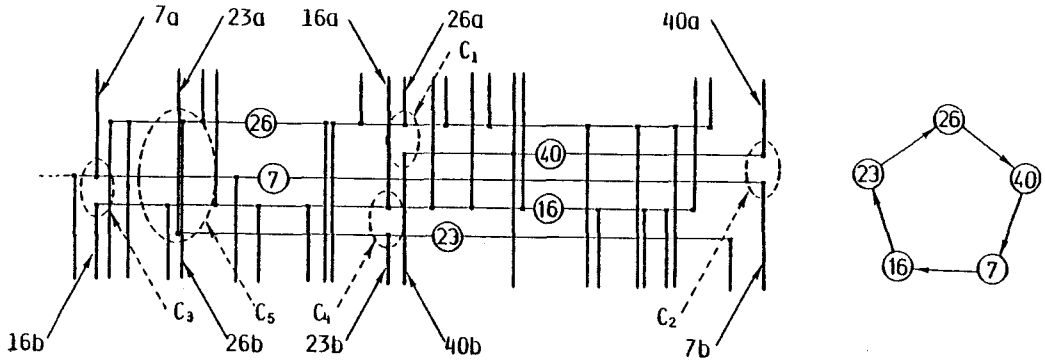
与えられた配線要求に対し幹線の位置を決定するために7.3節で述べた制約グラフを作成する。第7.2表は例題1および2の配線要求に対して作成された制約グラフの規模を示している。また、第7.5図は例題2の制約グラフを隣接行列の形で表したものである。第7.2表より、例題2の制約グラフは例題1の制約グラフに比べて節点数が少ないにもかかわらず、非常に多くのサイクルを含んでいることが分かる。これはグラフの虫喰度が例題1より例題2のほうが小さいことに起因するものである。また同表より、各制約グラフはサイクルが複雑に重なり合った結果、それぞれ大きさ15および19の強連結成分をもつことも分かる。このことは、例題1の場合は69個の配線要求のう

第7．2表　例題1および2の制約グラフの規模と半順序構造化結果

		例題1	例題2
規模	節点の数 : l	69	19
	枝の数 : n	79	120
	虫喰度 : $1-n/(l^2-1)$	0.983	0.649
	含まれる強連結成分の大きさ	15	19
	サイクルの数 : m	6	2534
結果	制約式の選択個数 : q	24	34
	イテレーション回数	1	2
	除去分割される節点	26	6,10,12
	演算時間［秒］	0.039	0.047

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	
2	0	0	0	1	1	1	1	1	0	1	1	0	0	0	1	1	1	1	1
3	0	0	0	0	1	1	0	1	0	0	0	0	1	0	1	1	1	0	0
4	1	0	1	0	1	1	0	1	1	0	0	1	0	0	1	1	1	1	1
5	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
6	1	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	0	1	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
8	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	1	0	0	0	0	1	0	1	0	1	0	1	1	0	0	1	0	0	0
10	0	0	1	1	0	1	1	0	0	0	0	0	1	0	0	1	1	1	1
11	1	0	1	1	1	1	1	1	1	0	0	0	0	1	1	0	1	1	1
12	1	1	0	0	1	1	1	1	0	1	0	0	1	0	0	0	1	0	1
13	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	1	1	0	0
14	0	0	1	0	1	1	1	1	1	1	0	0	0	0	1	1	0	1	0
15	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0
16	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
17	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	1	0	1	0	1	1	0	0	0	1	1	0	1	1	0	0	0
19	1	0	1	0	1	1	1	1	0	0	0	0	0	1	1	1	1	1	0

第７．５図　例題２の制約グラフに対応する隣接行列



第７．６図　例題１における配線不可能な例とそれに対応するサイクルd

ちの54個はこのままで配線可能であるが残りの15個が配線不可能であることを、一方、例題２の場合は19個の配線要求の総てがこのままでは配線不可能であることを意味している。

配線不可能な一例を第7.6図に示す。同図は例題１に含まれる６個のサイ

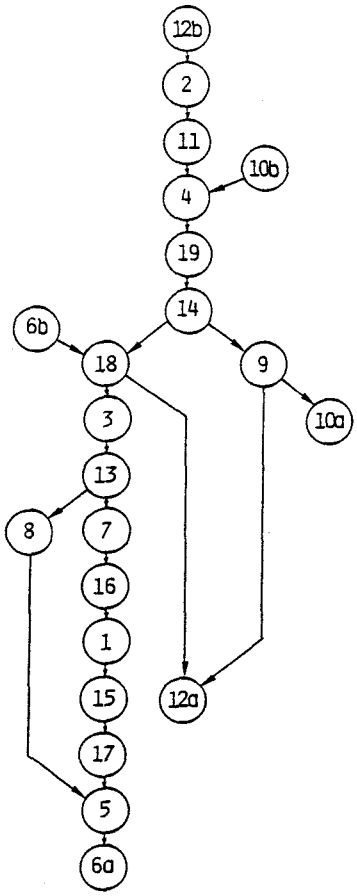
クルa～f(各配線がどのサイクルに含まれるかは第7.1表に示している)のうちの一つ、サイクルd に着目し、そのサイクルを構成している配線経路を第7.4(a)図から抜き出したものである。同図に描かれた支線のうち26aと40b、40aと７b、７aと16b、16aと23bがそれぞれ同じ垂直トラックを使用している(図中のC_1～C_4で表される部分)ことより、幹線は26、40、７、16、23の順に上から配置する必要がある。ところがC_5で表された部分に着目すると、23aと26bが同じ垂直トラックを使用しているため図のような支線の重複が生じ、このままでは配線不可能であることが分かる。

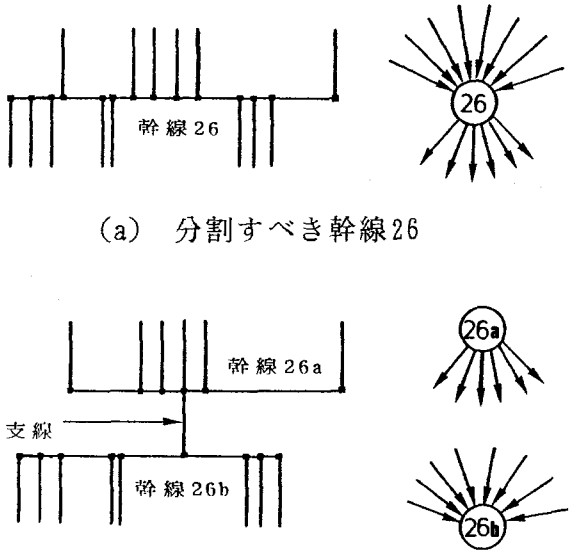
７．４．２　適用結果

与えられた総ての配線要求を満たす(100％配線率達成)ために、これらの制約グラフに半順序構造化アルゴリズムを適用して、強連結構造の半順序構造化を行う。前節でも述べたように、本適用例における半順序構造化は節点除去によるものであり、除去された節点に対応する幹線を分割すべき幹線と定めればよいことになる。なお、適用に際して目的関数は、幹線の分割処置を最小にするという意味で最少数の節点除去による半順序構造化問題と考え、その係数の値は総て１とした。

その結果、例題１の場合は節点26を除去分割することにより、一方、例題２の場合は節点６、10および12を除去分割することにより、それぞれの制約グラフの半順序構造化が可能になった。計算に要した演算時間およびアルゴリズムのイテレーション回数は第7.2表に示すとおりである。同表より、本アルゴリズムを用いればいくら多くのサイクルを含む強連結構造でも極めて短時間で容易に半順序構造化できることが分かる。第7.7図は例題２の半順序構造化結果を最少数の枝を用いた多階層有向グラフで表したものである。

これらの半順序構造化結果に基づき、多階層有向グラフ上のレベルが上の幹線から番号の若い水平トラックに互いに重ならないように順次詰めて配置していくことにより、各幹線の位置を決定することができる。この際、入力枝のみをもつ節点と出力枝のみをもつ節点(注7-1)に分割された幹線は、新





(a)　分割すべき幹線26

(b)　幹線26の分割

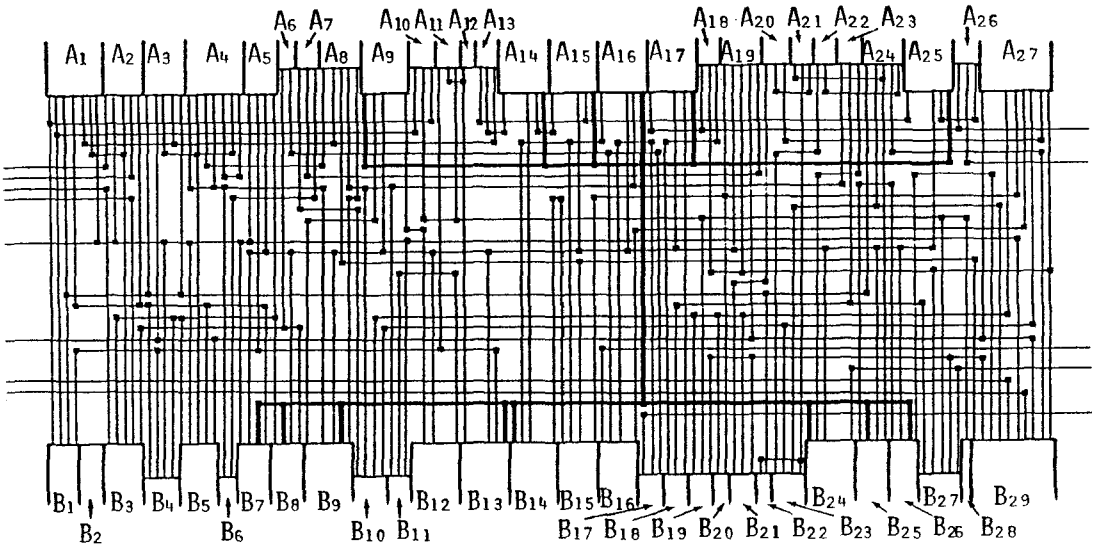
第７．８図　例題１の幹線26の分割方法

第７．７図　例題２の制約グラフの半順序構造化結果

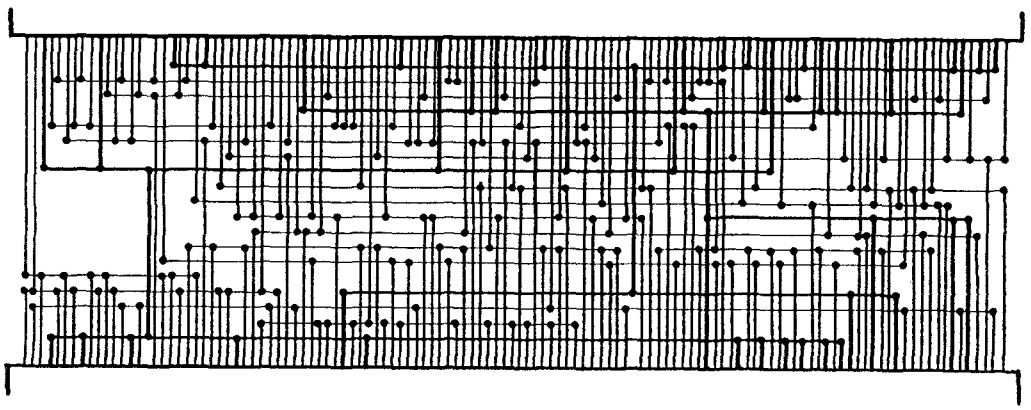
たに支線を設けてこれらを接続しておけばよい。[13] 第7.8図は例題１の幹線26の分割を示したものである。

以上のようにして設計された各例題の配線結果を第7.9図に示す。同図において、■印はスルーホールを表し、太線は分割された幹線をもつ配線経路を表している。同図より、総ての配線要求を満たすのに必要な水平トラック

(注7-1)　制約グラフ上で入力枝(出力枝)のみをもつ節点とは、換言すると下段(上段)のブロック列にある端子と接続する支線のみをもつ幹線に相当する。



(a) 例題１の配線結果



(b) 例題２の配線結果

第７．９図 配線結果

数は例題１の場合31本(配線領域内28本、ブロック列領域内３本)、例題２の場合は19本で十分であることが分かる。特に、例題１の本配線結果は文献(12)の配線結果(使用水平トラック数33本)に比べて、若干ではあるが水平トラック数の削減がなされており、チップ面積最小化の意味において優れていることを示唆している。

以上のように、半順序構造化アルゴリズムを用いればシステマティックに分割すべき幹線を決定することができ、しかも最も効率の良い、すなわち配線に必要な水平トラック数の増加が最小である分割を行うことができる。したがって、本アルゴリズムの適用は大規模でかつサイクルを多く含むような配線要求ほどその効力が倍加するものと考えられる。

７．５　結　言

本章では、半順序構造化アルゴリズムの実際問題への別の応用例として、ビルディングブロック方式ＬＳＩの配線問題を取り上げて、適用を行った。ここで得られた結果を要約すると、つぎの通りである。

（１）　ビルディングブロック方式ＬＳＩの幹線支線方式による配線設計において、実際の配線要求によっては配線経路間に同一垂直トラックの重複使用といったサイクル関係が発生して、要求どおりの配線が実現不可能になる事態が発生し得ることを明らかにした。そうして、このような事態の解消には節点除去による制約グラフの半順序構造化を行い、一部の幹線の分割処置を採る必要があることを示した。

（２）　つぎに、上述のような事態が実際に生じているＬＳＩ配線問題を例に取り上げて、半順序構造化アルゴリズムの適用により、本来の配線要求を満たすＬＳＩの配線設計を行った。ここで設計された配線は、従来の方法によるものと比較してＬＳＩの配線面積が小さくて済む点で優れており、本アルゴリズムのＬＳＩ集約化という意味での有効性を実証することができた。

第７章の参考文献

（１） 後藤 ： ＬＳＩレイアウト設計のＣＡＤ，電子通信学会誌，Vol.64，No.12，pp.1274-1281（昭 50）

（２） 川西,吉沢,可児 ： １次元ＭＯＳアレイにおけるゲート配列決定の一算法，電子通信学会論文誌，Vol.J59-A，No.2，pp.141-148（昭 51）

（３） W.N.Carr, et al. : MOS/LSI Design and Application, McGraw Hill (1972)

（４） K.Ueda, Y.Sugiyama and K.Wada : An Automatic Layout System for Masterslice LSI: MARC, IEEE Trans., Vol.SC-13, No.5, pp.716-721 (1978)

（５） 堀野,他 ： ＬＳＩ素子列の配線手法，電子通信学会 半導体トランジスタ研究会資料，SSD70-64（昭 46）

（６） 杉山,平野 ： ビルディングブロック方式ＬＳＩの自動配置配線設計システム，電子通信学会 半導体トランジスタ研究会資料，SSD71-23（昭 46）

（７） 橋本,西川 ： ＬＳＩ配線自動設計，電子通信学会誌，Vol.58，No.4，pp.370-375（昭 50）

（８） 吉田 ： ＬＳＩのレイアウト設計，電子通信学会誌，Vol.61，No.7，pp.737-743（昭 53）

（９） 小沢,堀野 ： ＭＯＳＬＳＩのレイアウトＣＡＤ，昭和48年電気四学会講演論文集，206（昭 48）

（10） 増田,藤井 ： 強連結構造の半順序構造化アルゴリズム―ＬＳＩ配線設計問題への応用―，計測自動制御学会 第10回システムシンポジウム，pp.275-280（昭 59）

（11） 増田,藤井 ： 節点分割による強連結構造の半順序構造化とその応用，計測自動制御学会論文集（投稿中）

（12） Y.Nakada : High Packing Density LSI Layout System with Interactive Facilities, Dig. ISSCC, Vol.17, pp.46-51 (1974)

(13) 浅野,他 : ビルディングブロック方式ＬＳＩの配線の実現可能性について, 電子通信学会論文誌, Vol.J56-A, No.9, pp.489-496 (昭 48)

第8章　結　　論

本論文では大規模強連結システムのための有効な構造分析手法を開発することを目的として、緩和法の概念を用いた半順序構造化アルゴリズムを提案し、種々の実際問題への適用を行った。本論文で得られた結果を要約するとつぎのようになる。

第2章では、システム構造分析の概要、その基礎理論、従来の代表的手法および問題点について概説した。その結果、大規模強連結システムの構造分析に関しては未だ十分な研究がなされてなく、実用に耐えうる有効な手法が確立されていないことを示唆した。

第3章では、本論文を通じて議論の対象とする強連結システムの半順序構造化問題に関して種々の予備的考察を行った。まず初めに、強連結構造の基本的概念を数学的に定義し、システム構造が強連結になる原因は構造内に存在する非推移関係(サイクル)にあることを明確にした。つぎに、こういった強連結システムの構造分析においては、構造内に存在する総てのサイクルを解消して半順序構造化することが極めて有用であることを明らかにした。さらに、強連結構造の具体的な半順序構造化の方法として、枝除去による方法と節点除去による方法の二通りの方法を示した。最後に、これら二通りの方法による半順序構造化を数理計画問題として定式化することを試みた。その結果、枝除去による半順序構造化は構造内に含まれる枝およびサイクルをそれぞれ変数および制約式としてもつ0-1線形計画問題に、一方、節点除去による半順序構造化は節点およびサイクルをそれぞれ変数および制約式としてもつ0-1線形計画問題に定式化することができた。

第4章では、第3章で定式化された強連結システムの半順序構造化問題を効率よく解くための手法として、緩和法の概念を導入した半順序構造化アルゴリズムを提案した。まず初めに、極めて多くのサイクルを含む強連結構造の半順序構造化問題を効率よく解くためには、総ての制約式を陽に考慮しなくてもよい緩和法の考え方が有効であることを明らかにした。つぎに、この緩和法の概念を導入して、強連結構造の半順序構造化アルゴリズムを新たに

開発した。これにより、構成要素数が100以上かつ数万のサイクルを含む大規模強連結構造の半順序構造化が極めて容易に行えるようになった。さらに、上記のアルゴリズムをより効率よく運用するために、部分問題の解法および部分問題に逐次追加する制約式の選択個数に関して検討を行った。その結果、時間複雑度が$O(\varepsilon mn^2)$で、非常に高い割合で最適解を得ることのできる、極めて効率のよい部分問題の近似解法を提案した。また、部分問題に逐次追加していく制約式の選択個数としては、構造内に含まれる枝の数の約1/4程度にその値を設定すると、半順序構造化アルゴリズムの演算効率が最も良くなることを実験的に明らかにした。

第5章、第6章および第7章では、第4章で開発した半順序構造化アルゴリズムを種々の実際問題に応用して、大規模強連結システムの構造分析における半順序構造化の意義を実用面から明確にするとともに、アルゴリズムの実用性および有効性の実証を行った。

まず第5章では、一対比較多数決を基礎としたランキング問題を取り上げて適用を行った。ここでは、まず初めに一対比較多数決に基づく意思決定者集団の選好構造は、「投票のパラドックス」や「個人の判断誤差」が原因して往々に強連結構造になることを、7名の試問委員の方々の御協力を得て、実際に口頭試問における学生間の順位付け実験を行うことにより実証した。つぎに、この実験で得られた7名の試問委員の集団選好構造に半順序構造化アルゴリズムを適用して、その中に含まれていた二つの強連結成分の半順序構造化を行った。その結果、一対比較多数決だけでは十分な順序付けのできなかった非推移的な学生間の順位関係もさらに詳細に順位付けすることができることを明らかにした。そうして、本アルゴリズムが集団意思決定過程における合意形成のための支援手法として有効であることを示した。

第6章では、プラントシミュレーションにおける計算手順決定問題を取り上げて半順序構造化アルゴリズムの適用を行った。まず最初に、大規模強連結構造をもつシステムのシミュレーション計算においては、一度に数値計算を行うことは規模の大きさあるいは非線形性が原因して極めて困難であり、通常、繰り返し計算によってシミュレーションを行わなければならないこと

を具体的に示した。そうして、この繰り返し計算を実行するためには、システムの一部のストリームを除去し構造を半順序構造化して、その計算手順を決定する必要があることを明確にした。つぎに、実際の接触式硫酸プラントモデルを取り上げて、半順序構造化アルゴリズムによりそのシミュレーション計算手順の決定を試みた。その結果、本適用例の場合、二つの最適計算手順が存在することが明らかになった。さらに、これらの手順に従ってシミュレーションを行えば、小規模な数値計算を繰り返すだけで容易にシミュレーション結果を得ることができることを示した。

最後に第７章では、ビルディングブロック方式ＬＳＩの配線問題を取り上げて、適用を行った。まず初めに、ビルディングブロック方式ＬＳＩの幹線支線方式による配線設計において、実際の配線要求によっては配線経路間に同一垂直トラックの重複使用といったサイクル関係が発生して、要求どおりの配線が実現不可能になる事態が発生し得ることを明らかにした。そうして、このような事態の解消には節点除去による制約グラフの半順序構造化を行い、一部の幹線の分割処置を採る必要があることを示した。つぎに、上述のような事態が実際に生じているＬＳＩ配線問題を例に取り上げて、半順序構造化アルゴリズムの適用により、本来の配線要求を満たすＬＳＩの配線設計を行った。ここで設計された配線は、従来の方法によるものと比較してＬＳＩの配線面積が小さくて済む点で優れており、本アルゴリズムのＬＳＩ集約化という意味での有効性を実証することができた。

謝　辞

本研究を行うに際し、終始御懇切な御指導ならびに温情ある御鞭撻を賜った大阪大学工学部電気工学科藤井克彦教授に深甚なる謝意を表す。

本論文の作成にあたり、電気工学科鈴木胖教授、木下仁志教授、山中千代衛教授、電子工学科児玉慎三教授、通信工学科中西義郎教授、健康体育部黒田英三教授には多くの有益な御助言を戴いた。ここに感謝の意を表す。

また、本研究遂行にあったて、種々の有益な御教示、御助言を戴いた精密工学科田村坦之助教授、電子工学科築山修治助手ならびに第5章の適用実験に御理解、御協力戴いた電気工学科森田龍彌助教授、平木昭夫助教授、白藤純嗣助教授、松浦虔士助教授、岩見基弘講師の各氏に深謝する。

さらに、赤沢堅造助手、田口英郎助手を始めとする電気工学科藤井研究室および鈴木研究室の諸兄には、本研究を通じて数々の有益な御助言、御助力を戴いた。ここにあらためて御礼申し上げる。

最後に、本学工学部あるいは工学研究科の卒業研究として本研究に参加され、よき共同研究者となって戴いた奥村信義君(現在、三菱電機勤務)、新山徹君(川崎製鉄勤務)、平峰正信君(大学院学生)、河村偉光君(シャープ勤務)、木島敏和君(学部学生)に衷心より感謝の意を表すともに厚く御礼申し上げる。

業績目録

Ⅰ．論　文　発　表

（1） T.Masuda and K.Fujii : A Method for Determining a Preference Solution of the Multiple Objective Problem, Proceedings of the IFAC Symposium on Environmental Systems Planning, Design and Control, Kyoto, pp.159-165 (1977 August)

（2） T.Masuda and K.Fujii : Application of Multiobjective Optimization Method to an Environmental System, Special Project Research on Environmental Pollution, Vol.4, pp.199-206 (1979 March)

（3） T.Masuda and K.Fujii : An Approach to Dynamic Optimal Control Problem with Multiple Criteria, Technology Reports of Osaka University, Vol.29-2, No.1500, pp.383-393 (1979 Octorber)

（4） T.Masuda and K.Fujii : Multiobjective Optimization Method and its Application to Dam Control Problem, 電子通信学会論文誌, Vol.E-62, No.11, pp.741-748 (昭54- 11)

（5） T.Masuda, N.Okumura and K.Fujii : A New Method for Discrete Multiobjective Decision Problem — Indifference Region Method (IRM), Technology Reports of Osaka University, Vol.32-2, No.1658, pp.49-56 (1982 Octorber)

（6） 増田・藤井 : 強連結システムの半順序構造化問題の定式化に関する一考察, 計測自動制御学会論文集, Vol.20, No.4, pp.364-366 (昭59-4)

（7） 増田・新山・藤井 ： 強連結システムの半順序構造化とその応用，システムと制御，Vol.28，No.8，pp.544-550（昭59- 8）

（8） 増田・新山・藤井 ： 緩和法の概念に基づく強連結システムの半順序構造化，計測自動制御学会論文集，Vol.20，No.9，pp.814-821（昭59-9）

（9） T.Masuda，T.Niiyama and K.Fujii ： Identification of Multiattribute Utility Function by GMDH，Technology Reports of Osaka University，Vol.34-2，No.1769，pp.247-253（1984 Octorber）

（10） 増田・藤井 ： 節点分割による強連結構造の半順序構造化とその応用，計測自動制御学会論文集（投稿中）

（11） 増田・藤井 ： 強連結となる一対比較結果の擬似区間順序化，計測自動制御学会論文集（投稿予定）

II．口　頭　発　表

（1） 増田・黒田・藤井 ： 状態に制約のある純慣性系の最短時間制御，昭和48年 電気学会全国大会，1314（昭48- 4）

（2） 増田・黒田・藤井 ： 大規模線形計画問題に対するLagrangeの双対性を用いた分解手法，日本自動制御協会 第18回学術講演会，L3（昭49-5）

（3） 増田・黒田・藤井 ： 双対性を利用した分解法の最適制御問題への適用，第17回自動制御連合講演会，1016（昭49- 11）

（4） 増田・藤井 ： ベクトル評価関数を考慮したパラメータ最適化，日本自動制御協会 第19回学術講演会，S3（昭50- 5）

（5） 増田・藤井 ： 多目的数理計画問題における妥協度最小化手法，電気学会 システム制御研究会資料，SC-76-12（昭51- 1）

（6） 増田・藤井 ： 最適制御問題の多目的評価について，計測自動制御学会 第6回制御理論シンポジウム，pp.169-174（昭52- 5）

（7） T.Masuda and K.Fujii : A Method for Determining a Preference Solution of the Multiple Objective Problem, IFAC Symposium on Environmental Systems Planning, Design and Control, pp.159-165 (1977 August)

（8） 増田・藤井 ： 多目的最適化手法の多目的ダム操作問題への適用，第21回自動制御連合講演会，3060（昭53- 11）

（9） 増田・藤井 ： 二次元トレードオフ曲線を用いた最適選好解決定手法と動的問題への応用，電気学会 システム制御研究会資料，SC-78-19（昭53- 12）

（10） 増田・藤井 ： 離散型多目的決定問題におけるパレート最適解集合の生成，電気学会 システム制御研究会資料，SC-79-29（昭54- 12）

（11） 増田・藤井 ： 意思決定者の無差別領域を用いた離散型多目的決定問題の選好解決定法，電気学会 システム制御研究会資料，SC-80-9（昭55- 3）

（12） 増田・藤井 ： 離散型多目的決定問題における意思決定者の無差別領

域の推定方法（その 1），昭和55年度 電子通信学会総合全国大会，60（昭55- 3）

(13) 増田・奥村・藤井 ：離散型多目的決定問題における意思決定者の無差別領域の推定方法（その 2），昭和55年度 電子通信学会総合全国大会，61（昭55- 3）

(14) 奥村・増田・藤井 ：GMDHを用いた多属性効用関数の同定，日本自動制御協会 第25回システムと制御研究発表講演会，M1（昭56- 5）

(15) 増田・奥村・藤井 ：AICを選択基準としたGMDHによる多属性効用関数の同定，昭和56年度 電子通信学会 情報・システム部門全国大会，30（昭56- 10）

(16) 増田・藤井 ：多層型GMDHによる多属性効用関数の同定，計測自動制御学会 第7回システムシンポジウム，pp.199-204（昭56- 10）

(17) 奥村・増田・藤井 ：GMDHを用いた多属性効用関数の同定（第２報），第24回自動制御連合講演会，1066（昭56- 11）

(18) 増田・新山・藤井 ：GMDHによる多属性効用関数同定のための対話型アルゴリズム，日本自動制御協会 第26回システムと制御研究発表講演会，Q1（昭57- 5）

(19) 増田・新山・藤井 ：GMDHを用いた多属性効用関数の同定（第3報）＝同定実験による検討＝，昭和58年度電子通信学会総合全国大会，1312（昭58- 4）

(20) 増田・平峰・藤井 ：パラメトリックISM法による集団選好の構造化，

日本自動制御協会 第27回システムと制御研究発表講演会，W7（昭58-5）

(21) 増田・藤井 ：強連結構造の半順序化，計測自動制御学会 第9回システムシンポジウム，pp.297-302（昭58- 8）

(22) 増田・新山・藤井 ：緩和法を用いた強連結システムの半順序構造化，第26回自動制御連合講演会，1066（昭58- 11）

(23) 増田・藤井 ：緩和法の概念に基づく強連結システムの半順序化とその応用，計測自動制御学会 第2回知識工学シンポジウム，pp.45-50（昭59- 3）

(24) 増田・藤井 ：強連結構造の半順序化アルゴリズムの開発，日本自動制御協会 第28回システムと制御研究発表講演会，S2（昭59- 5）

(25) 増田・新山・平峰・藤井 ：強連結構造の半順序化アルゴリズムのランキング問題への応用，日本自動制御協会 第28回システムと制御研究発表講演会，S3（昭59- 5）

(26) 増田・新山・藤井 ：強連結構造の半順序化アルゴリズムの計算手順決定問題への応用，日本自動制御協会 第28回システムと制御研究発表講演会，S4（昭59- 5）

(27) 増田・藤井 ：強連結構造の半順序構造化アルゴリズム―ＬＳＩ配線設計問題への応用―，計測自動制御学会 第10回システムシンポジウム，(昭59-8)

(28) 平峰・増田・藤井 ：ＬＳＩの機能ブロック配置問題への半順序構造

化アルゴリズムの適用，第27回自動制御連合講演会，1096（昭59-11）

Ⅲ．その他

（1） 増田・藤井 ：多目的数理計画問題に対する最適化手法―妥協度最小化手法―，科学研究特定（1）環境汚染の検知と制御 研究報告書(IV)，pp.275-282（昭51-2）

（2） 増田・藤井 ：環境問題における選好解決定法について，科学研究特定（1）環境汚染の検知と制御 研究報告書(IV)，pp.213-216（昭52-2）

（3） 増田・藤井 ：多目的ダム操作問題の数学モデル，科学研究特定（1）環境汚染の検知と制御 研究報告書(IV)，pp.256-259（昭53-2）

付　　録

第４章の緩和法の概念に基づく半順序構造化アルゴリズム(枝除去による場合)のＦＯＲＴＲＡＮプログラムを付録として付記しておく。

本プログラムは節点数l＝30、枝数n＝450の強連結構造まで適用可能なように作成されている。したがって、これ以上の規模の強連結構造を対象とする場合には、プログラム中のＤＩＭＥＮＳＩＯＮ文などを再調整する必要がある。また、本プログラムでは半順序構造化アルゴリズム中のサイクル列挙法としてはJohnsonのアルゴリズムを、部分問題P^kの解法としては4.4.1の近似解法をそれぞれ採用している。

Ⅰ．パラメータおよびデータの入力方法

プログラムの実行に際して、パラメータおよびデータをつぎの順序で入力しなければならない。

i) N　　：強連結構造の節点数 l。

　IPRINT　　：各イテレーションで抽出したサイクルをプリントアウトするか否かの指定。 列挙する場合「１」、しない場合「０」とする。

ii) ADJO(I,J)　：強連結構造を表現した隣接行列。一行ごと入力する。

iii) WEIGHT(I,J)：隣接行列の各枝に付与された重み係数を表現した重み行列。一行ごと入力する。

iv) IJ　　：「１」に設定する。

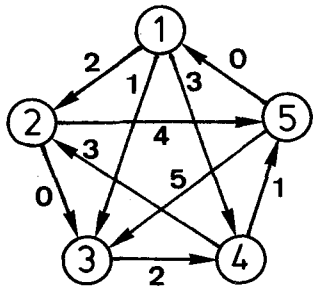
　ISELECT　　：各イテレーションで部分問題P^kに追加する制約式(抽出するサイクル)の選択個数 q。構造内の総てのサイクルを抽出して一挙に解く場合は「－１」に設定する。

　IRANGE　　：部分問題P^kの近似解法における解の探索範囲を指定するパラメータε。通常「０」に設定する。

　IADD　　：「０」に設定する。

Ⅱ．入出力例

第A1図に示す節点数l＝５、枝数n＝10の強連結構造(図中の枝の横の数値はその枝に付与された重みを表している)を例にとり、本プログラムを実行して最適な半順序構造化結果を得るまでの入出力例を以下に示す。



(a) 強連結構造グラフ

$$A = \begin{array}{c} \\ ① \\ ② \\ ③ \\ ④ \\ ⑤ \end{array} \begin{array}{c} \begin{array}{ccccc} ①&②&③&④&⑤ \end{array} \\ \begin{bmatrix} 0&1&1&1&0 \\ 0&0&1&0&1 \\ 0&0&0&1&0 \\ 0&1&0&0&1 \\ 1&0&1&0&0 \end{bmatrix} \end{array}$$

(b) 隣接行列A

$$W = \begin{array}{c} \\ ① \\ ② \\ ③ \\ ④ \\ ⑤ \end{array} \begin{array}{c} \begin{array}{ccccc} ①&②&③&④&⑤ \end{array} \\ \begin{bmatrix} 0&2&1&3&0 \\ 0&0&0&0&4 \\ 0&0&0&2&0 \\ 0&3&0&0&1 \\ 0&0&5&0&0 \end{bmatrix} \end{array}$$

(c) 重み行列W

第Ａ１図　例題に使用した強連結構造

【入力部分】

```
VERTEX (PART) ,   1 FOR PRINT CYCLES
5  0
ADJACENCY MATRIX   (PART MATRIX)
0 1 1 1 0
0 0 1 0 1
0 0 0 1 0
0 1 0 0 1
1 0 1 0 0
WEIGHT MATRIX   (PART MATRIX)
0 2 1 3 0
0 0 0 0 4
0 0 0 2 0
0 3 0 0 1
0 0 5 0 0
SCALE ,  SELECT-NUMBER ( 'Q' )0 ) ,  SEEKING-AREA ( L ) ,  (L=0)-OPERATE FOR 1
  1       7       0       0
```

節点数とサイクルをプリントアウトするか否かを指定する。

隣接行列を入力する。

重み行列を入力する。

各パラメータを設定する。

【出力部分】

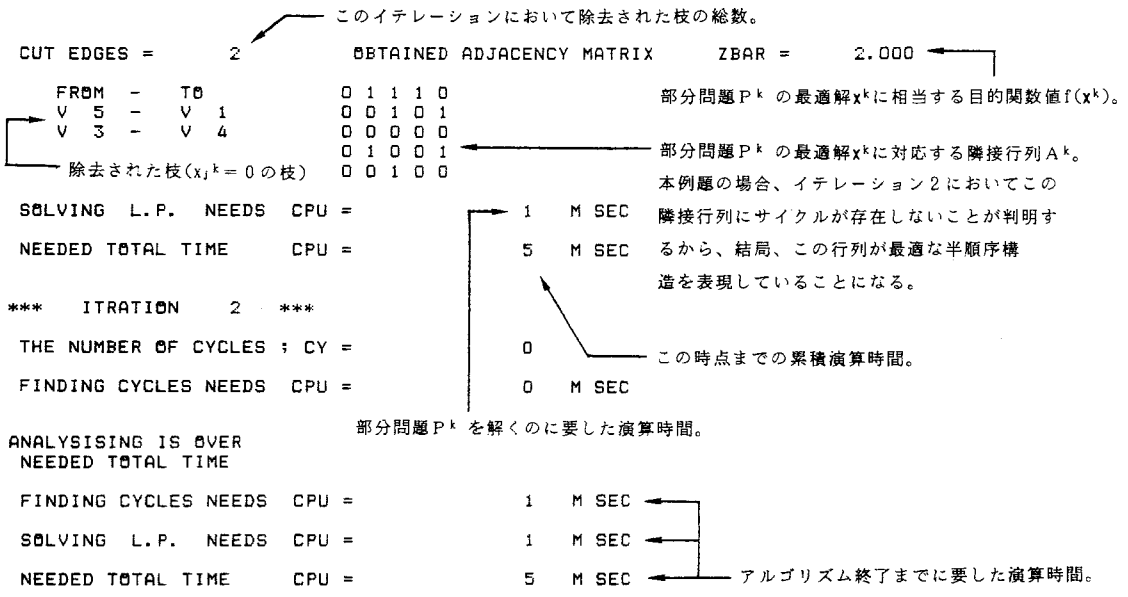
```
*****  DIRECTED GRAPH ANALYSIS  *****
   THE  NUMBER  OF  VERTICES  N =    5
   THE  NUMBER  OF  EDGES     E =   10

  ***   ITERATION   1   ***
   THE NUMBER OF CYCLES ; CY =          7
   FINDING CYCLES NEEDS  CPU =          1   M SEC
   CYCLE MATRIX (COEF.)        7      (STEP 4  - 0)
```

このイテレーションにおいて抽出されたサイクルの個数。

サイクル抽出に要した演算時間。

この値が「０」の場合、アルゴリズムのＳtep ６において$L^{k+1} = L^k \cup V^k - D^k$によって、一方、「１」の場合は$L^{k+1} = L^k \cup V^k$によって、つぎの部分問題$P^{k+1}$の制約式集合$L^{k+1}$が作成されたことを意味している。

このイテレーションにおいて解かれる部分問題P^kの制約式の総数。



Ⅲ. プログラムリスト

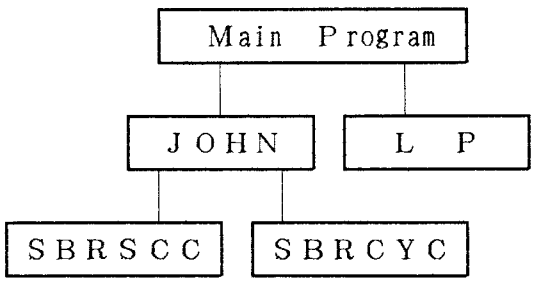
本プログラムは第A2図に示すように1つのメイン・プログラムと4つのサブルーチン・プログラムから構成されている。各プログラムのもつ機能はつぎのとおりである。

◎メイン・プログラム

隣接行列、重み行列などを読み込んで、4.3節のアルゴリズムに従い、枝除去による最適な半順序構造を求める。

◎サブルーチン　JOHN

隣接行列A^kに含まれるサイクルを抽出する。



第A2図　プログラムの構成

◎サブルーチン　SBRSCC および SBRCYC

サブルーチンJOHNに従属していて、サイクル抽出のためのBlocking-unblocking操作、新たな強連結成分の探索などを行う。

◎サブルーチン　ＬＰ

部分問題P^kの最適解x^kを近似解法によって求める。

以下に、これらの具体的なプログラムリストを示す。

【メイン・プログラム】

```
0010C         WEIGHT & ADJ-(41)   OPERATION-(5)      SOLUTION-(42)
0020C
0030C*****      DIRECTED GRAPH  ANALYSIS      *****
0040C
0050C           MINIMUM FEEDBACK ARC SET  OBTAINED
0060C
0070C      **  PARAMETER  **
0080C      N       IS  THE NUMBER OF VERTICES  OF  GIVEN DIGRAPH
0090C      M       IS  THE LENGTH OF ADJACENCY STRUCTURE  A(M) ,  I.E. ,
0100C         ;  THE NUMBER OF EDGES  PLUS  THE NUMBER OF VERTICES  OF  DIGRAPH
0110C      AI(N)   IS  A SET OF POINTERS FOR EACH ADAJACENCY LIST
0120C      A(M)    IS  THE SET OF ADJACENCY LISTS OF GIVEN DIGRAPH
0130C
0140C      IN ANY ADJACENCY LIST OF VERTEX  I   ( A(AI(I) - A(AI(I+1)-1) )
0150C         ALL VERTICES ARE ARRANGED IN INCREASING ORDER
0160C         AND "0" IS ATTACHED AT THE END OF LIST AS  END-MARKER
0170C
0180       IMPLICIT INTEGER (A,C,I-N)
0190       INTEGER         EDGE
0200       REAL CYCLEM
0210       DIMENSION ADJ(30,30),   AID(30), NNS(450),WEIGHT(30,30)
0220       DIMENSION ADJO(30,30),   BB(430),FCOEF(450),BBS(430),WEIGHTO(30,30)
0230       DIMENSION CUT(450),IFROMV(450),ITOV(450)
0240       COMMON /ALIST/ N,M,AI(31),A(480)
0250       COMMON /CYCLE/ IS1,CY,ITJ1,ITJ2,CE,EDGE,ISELECT
0260       COMMON /STOCK/ CYCLEM(430,450),INDEXM(30,30),   LENG(430),IPRINT
0270       COMMON /TIMEB/ ITB1,ITB2
0280C
0290       CALL FPARAM(1,132)
0300       CALL LONGIO(6,132)
0310C
0320C      **  INPUT  **
0330C
0340       WRITE(6,100)
0350   100 FORMAT(2X,"VERTEX (PART) ,  1 FOR PRINT CYCLES")
0360       READ(5,LIST) N,IPRINT
0370       WRITE(6,99)
0380    99 FORMAT(2X,"ADJACENCY MATRIX  (PART MTRIX) ")
0390       READ(41,LIST)((ADJO(I,J),J=1,N),I=1,N)
0400       WRITE(6,101)
0410   101 FORMAT(2X,"WEIGHT MATRIX   (PART MATRIX) ")
0420       READ(41,LIST)((WEIGHTO(I,J),J=1,N),I=1,N)
0430       WRITE(6,102)
0440   102 FORMAT(2X,"SCALE , SELECT-NUMBER ( 'O' )O ) , SEEKING-AREA ( L ) ,",
0450      &          " (L=O)-OPERATE FOR  1" )
0460       READ (5,LIST) IJ,ISELECT,IRANGE,IADD
0470C
0480C      **  TEST GRAPH  MAKING  **
0490C
0500       DO 300 I=1,N
0510       DO 300 J=1,N
0520   300 WEIGHT(I,J)=WEIGHTO(I,J)
0530       IF( IJ.EQ.1 ) GOTO 301
0540       DO 302 K1=2,IJ
0550       K2=N*(K1-1)
0560       DO 303 I=1,N
0570       DO 303 J=1,N
0580       WEIGHT( I+K2,J+K2 )=WEIGHT(I,J)
```

```
0590       ADJO( I+K2,J+K2 )=ADJO(I,J)
0600   303 CONTINUE
0610       WEIGHT(K2,K2+1)=0
0620       WEIGHT(K2+1,K2)=0
0630       ADJO(K2,K2+1)=1
0640       ADJO(K2+1,K2)=1
0650   302 CONTINUE
0660   301 N=N*IJ
0670C
0680       CALL CPTIME(ITT1)
0690C
0700C      **  INITIALIZATION  **
0710C
0720       CE=0
0730       ITE=0
0740       ZZBAR=-1.0E10
0750       ITIMEJ=0
0760       ITIMEB=0
0770C
0780C      **  MAKING  ADJACENCY-MATRIX  &  INDEX-MATRIX (SMALLER,YOUNGER)  **
0790C
0800       EDGE=0
0810       DO 15 I=1,N
0820       DO 15 J=1,N
0830       IF( ABS( ADJO(I,J) ) .LT. 1.0E-10 ) GOTO 15
0840       EDGE=EDGE+1
0850       ADJ(I,J)=1
0860       IFROMV(EDGE)=I
0870       ITOV(EDGE)=J
0880       FCOEF(EDGE)=-WEIGHT(I,J)
0890    15 CONTINUE
0900C
0910       DO 45 I=1,EDGE-1
0920       IFL=I
0930       DO 46 J=I+1,EDGE
0940       IF( ABS( FCOEF(IFL) ) .GT. ABS( FCOEF(J) ) ) IFL=J
0950    46 CONTINUE
0960       IF( IFL.EQ.I ) GOTO 45
0970       F1=FCOEF(I)
0980       FCOEF(I)=FCOEF(IFL)
0990       FCOEF(IFL)=F1
1000       M1=IFROMV(I)
1010       IFROMV(I)=IFROMV(IFL)
1020       IFROMV(IFL)=M1
1030       M1=ITOV(I)
1040       ITOV(I)=ITOV(IFL)
1050       ITOV(IFL)=M1
1060    45 CONTINUE
1070       DO 47 I=1,EDGE
1080       INDEXM( IFROMV(I),ITOV(I) )=I
1090    47 CONTINUE
1100C
1110       WRITE(6,103) N,EDGE
1120   103 FORMAT(//2X,"*****  DIRECTED GRAPH ANALYSIS  *****"//
1130      &          5X,"THE  NUMBER  OF  VERTICES  N = ",I4//
1140      &          5X,"THE  NUMBER  OF  EDGES     E = ",I4)
1150C            WRITE(6,109)
1160C        109 FORMAT(/5X,"INDEX MATRIX  ( EDGE OF 'VERTEX-I TO VERTEX-J')" /)
1170C            DO 70 I=1,N
1180C            WRITE(6,110) (INDEXM(I,J),J=1,N)
1190C        110 FORMAT(5X,25I4)
1200C         70 CONTINUE
1210C
1220C      **  ITERATION  **
1230C
1240 1000 ITE=ITE+1
1250       WRITE(6,113) ITE
1260   113 FORMAT(//4X,"***   ITRATION ",I4,"   ***" )
1270C
1280C      **  MAKING   ADJACENCY-LIST FOR JOHNSON'S  **
1290C
1300       DO 10 I=1,N
```

```
1310       AI(I)=0
1320       AID(I)=0
1330    10 CONTINUE
1340       M=0
1350       DO 20 I=1,N
1360       DO 30 J=1,N
1370       IF( ADJ(I,J) .LT. 1 ) GOTO 30
1380       M=M+1
1390       AID(I)=AID(I)+1
1400       A(M)=J
1410    30 CONTINUE
1420       M=M+1
1430       AID(I)=AID(I)+1
1440       A(M)=0
1450    20 CONTINUE
1460       AI(1)=1
1470       DO 40 I=2,N
1480       AI(I)=AI(I-1)+AID(I-1)
1490    40 CONTINUE
1500       AI(N+1)=M+1
1510C
1520C            WRITE(6,104)
1530C         104 FORMAT(/5X,"ADJACENCY STRUCTURE    ( ADJACENCY LIST )"/)
1540C            DO 50 I=1,N
1550C            IS=AI(I)
1560C            IT=AI(I+1)-1
1570C            WRITE(6,105) I,( A(J),J=IS,IT )
1580C         105 FORMAT(8X,"ADJ(",I3,") : "4X,25I4,(/22X,25I4) )
1590C          50 CONTINUE
1600C
1610       IF( IPRINT.NE.1 ) GOTO 61
1620       WRITE(6,106)
1630    106 FORMAT(/5X,"THE FOUND CYCLES       ( BY SEQUENTIAL VERTICES )"/)
1640     61 CONTINUE
1650C
1660       CALL JOHN
1670C
1680       WRITE(6,107) CY
1690    107 FORMAT(/5X,"THE NUMBER OF CYCLES ; CY =",I15)
1700       WRITE(6,108) ITJ2-ITJ1
1710    108 FORMAT(/5X,"FINDING CYCLES NEEDS  CPU =",I15,"   M SEC")
1720C
1730       IF( CY.EQ.0 ) GOTO 240
1740C
1750       ITIMEJ=ITIMEJ+(ITJ2-ITJ1)
1760C
1770C
1780C      **  PREPARING FOR "BALAS"  **
1790C
1800       DO 200 I=CE+1,CE+CY
1810       DO 201 J=1,EDGE
1820    201 CYCLEM(I,J)=-CYCLEM(I,J)
1830       BB(I)=FLOAT( LENG(I)-1 )
1840    200 CONTINUE
1850C
1860    250 IF( IADD.EQ.2 ) IADD=0
1870C
1880       WRITE(6,111) CE+CY,ISTEP4
1890    111 FORMAT(/5X,"CYCLE MATRIX (COEF.) ",I10,5X," (STEP 4  - ",I1,")"/  )
1900C            WRITE(6,114)(I,I=1,EDGE)
1910C         114 FORMAT(5X,"BB(I)   ",40I3)
1920C            DO 80 I=1,CE+CY
1930C            WRITE(6,112) I,BB(I),(CYCLEM(I,J),J=1,EDGE)
1940C         112 FORMAT(1X,I3,":",F5.0,2X,40F3.0)
1950C          80 CONTINUE
1960C
1970C      **  SOLVING LINEAR PROGRAM WITH 0-1 VARIABLES  **
1980C
1990       CALL LP(CE+CY,EDGE,CYCLEM,BB,FCOEF,IRANGE,  NNS,BBS,ZBAR)
2000C
2010C      **  OBTAINED ADJACENCY MATRIX  INTO  ADJ(I,J)  **
2020C
```

```
      DO 211 I=1,N
      DO 211 J=1,N
  211 ADJ(I,J)=ADJO(I,J)
      CUTN=0
      DO 210 I=1,EDGE
      IF( NNS(I).NE.0 ) GOTO 210
      CUTN=CUTN+1
      CUT(CUTN)=I
      ADJ( IFROMV(I),ITOV(I) )=0
  210 CONTINUE
C
C     **  CONSTRUCTING  CONSTRAINT EQUATIONS  **
C
      ISTEP4=1
      IF( ZBAR.LE.ZZBAR ) GOTO 230
      ZZBAR=ZBAR
      ISTEP4=0
  230 IFL=0 ; CE2=0
      DO 231 K=1,CE+CY
      IF( ISTEP4 .EQ. 0  .AND.  BBS(K) .GT. 0.0 ) GOTO 232
      IFL=IFL+1
      BB(IFL)=BB(K)
      DO 233 J=1,EDGE
  233 CYCLEM(IFL,J)=CYCLEM(K,J)
      GOTO 231
  232 CE2=CE2+1
      IF( (CE+CY+CE2).GE.300 ) GOTO 235
      IF( IADD.NE.1 ) GOTO 231
      BB(CE+CY+CE2)=BB(K)
      DO 234 J=1,EDGE
  234 CYCLEM(CE+CY+CE2,J)=CYCLEM(K,J)
      GOTO 231
  235 IADD=0
  231 CONTINUE
      CE3=CE+CY
      CE=IFL
C
      CALL CPTIME(ITT2)
C
      WRITE(6,115) CUTN,ZBAR
  115 FORMAT(/5X,"CUT EDGES =",I7,9X,"OBTAINED ADJACENCY MATRIX",
     &       5X,"ZBAR =",F10.3/)
      WRITE(6,116) ( ADJ(1,J),J=1,N )
  116 FORMAT(8X,"FROM  -   TO",10X,50I2)
      DO 220 I=1,CUTN
      ID=I
      IF( ID.GT.(N-1) ) GOTO 221
      WRITE(6,117) IFROMV( CUT(I) ),ITOV( CUT(I) ),( ADJ(I+1,J),J=1,N )
  117 FORMAT(8X,"V",I3,"  -   V",I3,8X,50I2)
  220 CONTINUE
      IF( CUTN.EQ.(N-1) ) GOTO 224
      GOTO 225
  221 WRITE(6,121) ( IFROMV( CUT(K) ),ITOV( CUT(K) ) , K=ID,CUTN )
  121 FORMAT(8X,"V",I3,"  -   V",I3/10(8X,5("V",I3,"  -   V",I3,8X) /) )
      GOTO 224
  225 DO 223 I=CUTN+2,N
      WRITE(6,118) (ADJ(I,J),J=1,N)
  118 FORMAT(30X,50I2)
  223 CONTINUE
C
  224 CONTINUE
      ITIMEB=ITIMEB+ITB2
      WRITE(6,119) ITB2
  119 FORMAT(/5X,"SOLVING  L.P.  NEEDS  CPU =",I15,"   M SEC")
      ITIMET=(ITT2-ITT1)
      WRITE(6,120) ITT2-ITT1
  120 FORMAT(/5X,"NEEDED TOTAL TIME     CPU =",I15,"   M SEC")
C
      GOTO 1000
C
C     **  SEARCING ADDED (IADD=1)  OR NOT  **
C
```

```
2750   240 IF( IADD.NE.1 ) GOTO 241
2760       L=0 ; IRANGE=0
2770       XL=(1.0-( FLOAT(L)/FLOAT(EDGE) ) )*100.
2780       WRITE(6,130) L,XL
2790   130 FORMAT(//4X,"FOLLOWING - SEARCHING IN  L.P.      L =",I5,
2800      &          10X,"( ",F6.1," % )" )
2810C
2820       IFL=0
2830       DO 242 I=1,CE2
2840       IFL=IFL+1
2850       BB(CE+IFL)=BB(CE3+IFL)
2860       DO 243 J=1,EDGE
2870   243 CYCLEM(CE+IFL,J)=CYCLEM(CE3+IFL,J)
2880   242 CONTINUE
2890       IADD=2
2900       CE=CE+CE2
2910       CY=0
2920       ISTEP4=0
2930       GOTO 250
2940C
2950C      **  END  OF  ANALYSISING  **
2960C
2970   241 CONTINUE
2980       WRITE(6,141)
2990   141 FORMAT(//4X,"ANALYSISING IS OVER"/5X,"NEEDED TOTAL TIME" )
3000       WRITE(6,108) ITIMEJ
3010       WRITE(6,119) ITIMEB
3020       WRITE(6,120) ITIMET
3030       DO 1002 I=1,N
3040C      WRITE(42,1003)(ADJ(I,J),J=1,N)
3050C1003 FORMAT(2X,40I3)
3060 1002 CONTINUE
3070C
3080       STOP
3090       END
```

【サブルーチン　ＪＯＨＮ】

```
3100C
3110C
3120C*****************************************************************************
3130       SUBROUTINE JOHN
3140C
3150C      MAIN PROCEDURE OF JOHNSON'S ALGORITHM FOR GENERATING ALL THE CYCLES
3160C
3170       IMPLICIT INTEGER (A-Z)
3180       REAL CYCLEM
3190       COMMON /ALIST/ N,M,AI(31), A(480)
3200       COMMON /CYCLE/ S,CY,T1,T2,CE,EDGE,SELECT
3210       COMMON /SCCMP/ VKC,VKMAX,CN,NCN,VK(30), SCC(30)
3220       COMMON /CRRNT/ BI(30), BP(30), STATE(30)
3230       COMMON /WORKS/ S1(30), S2(30), PTR(30), VCTR(30), L1(900),  L2(900)
3240       COMMON /CLIST/ CI(30), CA(900)
3250       COMMON /STOCK/ CYCLEM(430,450),INDEXM(30,30),  LENG(430),IPRINT
3260C
3270       CY=0
3280       CALL CPTIME(T1)
3290C
3300C      **  INITIALIZATIONS  **
3310C
3320       DO 10 I=1,N
3330       CI(I)=AI(I)
3340       BI(I)=0
3350    10 CONTINUE
3360       DO 20 I=1,M
3370       W=A(I)
3380       IF( W.NE.0 ) BI(W)=BI(W)+1
3390    20 CONTINUE
3400       TOP=1
```

```
      DO 30 I=1,N
      T=BI(I)
      BI(I)=TOP
      TOP=TOP+T
      SCC(I)=0
      VK(I)=I
   30 CONTINUE
      CN=0
      NCN=1
      S=1
      VKC=1
      VKMAX=N
C
C     **  DIVIDING DIGRAPH INTO STRONGLY CONNECTED COMPONENT  **
C
  100 CONTINUE
      CALL SBRSCC
C
C     **  CONSTRUCTING AN ADJACENCY STRUCTURE OF THE STRONGLY CONNECTED
C           COMPONENT CONTAINING CYCLE START VERTEX S  **
C
  200 CONTINUE
      VKMAX=0
      CN=SCC(S)
      DO 210 V=S,N
      IF( SCC(V).NE.CN ) GOTO 210
      VKMAX=VKMAX+1
      VK(VKMAX)=V
  210 CONTINUE
      IF( VKMAX.NE.1 ) GOTO 300
      S=S+1
      IF( S.LT.N ) GOTO 200
      GOTO 500
C
  300 CONTINUE
      DO 400 I=1,VKMAX
      V=VK(I)
      STATE(V)=0
      BP(V)=BI(V)
C
      CT=CI(V)
      T=AI(V)
  410 CONTINUE
      W=A(T)
      IF( W.EQ.0 ) GOTO 430
      T=T+1
      IF( W.LT.S ) GOTO 420
      IF( SCC(W).NE.CN ) GOTO 410
      CA(CT)=W
      CT=CT+1
      GOTO 410
  420 CONTINUE
      AI(V)=T
      GOTO 410
  430 CONTINUE
      CA(CT)=0
C
  400 CONTINUE
C
C     **  FINDING ALL THE CYCLES CONTAINING VERTEX S AND CONTAINED
C           IN THE STRONGLY CONNECTED SUBGRAPH INDUCED BY "VK"  **
C
      CALL SBRCYC
C
      S=S+1
      IF( S.GE.N ) GOTO 500
      VKC=2
      GOTO 100
C
C     **  EXIT OF THIS ROUTINE  **
C
```

```
4120  500 CONTINUE
4130      CALL CPTIME(T2)
4140C
4150      RETURN
4160      END
```

【サブルーチン ＳＢＲＳＣＣ】

```
4170C
4180C
4190C*****************************************************************************
4200      SUBROUTINE SBRSCC
4210C
4220      IMPLICIT INTEGER (A-Z)
4230      COMMON /ALIST/ N,M,AI(31), A(480)
4240      COMMON /CYCLE/ S,CY,T1,T2,CE,EDGE,SELECT
4250      COMMON /SCCMP/ VKC,VKMAX,CN,NCN,VK(30), SCC(30)
4260      COMMON /WORKS/ RCS(30), SCS(30), AP(30), STATE(30),
4270     &               NO(900),  LOWL(900)
4280C
4290C     **  INITIALIZATIONS  **
4300C
4310      K=0
4320      L=0
4330      DO 10 I=VKC,VKMAX
4340      W=VK(I)
4350      STATE(W)=0
4360   10 CONTINUE
4370C
4380C     **  FINDING  NEW ROOT VERTEX R  **
4390C
4400  100 CONTINUE
4410      IF( VKC.GT.VKMAX ) RETURN
4420      R=VK(VKC)
4430      IF( SCC(R).EQ.CN ) GOTO 200
4440  150 CONTINUE
4450      VKC=VKC+1
4460      GOTO 100
4470C
4480C     **  DIVIDING THE SUBGRAPH INDUCED BY THE SET OF VERTICES,
4490C             WHICH WERE CONTAINED IN THE STRONGLY CONNECTED COMPONENT
4500C             WITH INDEX "CN"  AND  ARE REACHED FROM ROOT VERTEX R,
4510C             INTO STRONGLY CONNECTED COMPONENTS  **
4520C
4530  200 CONTINUE
4540      V=R
4550C
4560C     **  STACKING THE MOST RECENTLY REACHED VERTEX V  **
4570C
4580  250 CONTINUE
4590      K=K+1
4600      RCS(K)=V
4610      L=L+1
4620      SCS(L)=V
4630      CNT=CNT+1
4640      NO(V)=CNT
4650      LOWL(V)=CNT
4660      STATE(V)=1
4670      AP(V)=AI(V)
4680C
4690C     **  EXPLOSING OF THE VERTICES ADJACENT TO VERTEX V  **
4700C
4710  300 CONTINUE
4720      T=AP(V)
4730      W=A(T)
4740      T=T+1
4750      AP(V)=T
4760      IF( W.EQ.0 ) GOTO 400
4770      IF( W.GE.S ) GOTO 320
4780      AI(V)=T
4790      GOTO 300
```

```
4800  320 CONTINUE
4810      IF( SCC(W)  .NE.CN ) GOTO 300
4820      IF( STATE(W).EQ. 0 ) GOTO 350
4830      IF( STATE(W).EQ. 2 ) GOTO 300
4840      IF( NO(W)  .GE. NO(V) ) GOTO 300
4850      IF( LOWL(V).GE. NO(W) ) LOWL(V)=NO(W)
4860      GOTO 300
4870C
4880C     **  EXTENDING TO VERTEX W  **
4890C
4900  350 CONTINUE
4910      V=W
4920      GOTO 250
4930C
4940C     **  BACKTRACKING  **
4950C
4960  400 CONTINUE
4970      IF( LOWL(V).EQ.NO(V) ) GOTO 500
4980      K=K-1
4990      W=RCS(K)
5000      IF( LOWL(W).GT.LOWL(V) ) LOWL(W)=LOWL(V)
5010      V=W
5020      GOTO 300
5030C
5040C     **  A NEW STRONGLY CONNECTED COMPONENT  IS  FOUND  **
5050C
5060  500 CONTINUE
5070      IF( L.EQ.0 ) GOTO 600
5080      W=SCS(L)
5090      IF( NO(W).LT.NO(V) ) GOTO 600
5100      L=L-1
5110      STATE(W)=2
5120      SCC(W)=NCN
5130      GOTO 500
5140C
5150  600 CONTINUE
5160      NCN=NCN+1
5170      K=K-1
5180      IF( K.EQ.0 ) GOTO 150
5190      V=RCS(K)
5200      GOTO 300
5210C
5220      END
```

【サブルーチン ＳＢＲＣＹＣ】

```
5230C
5240C
5250C*****************************************************************************
5260      SUBROUTINE SBRCYC
5270C
5280C     SUBROUTINE FOR GENERATING ALL THE CYCLES CONTAINING VERTEX S
5290C            AND CONTAINED IN A STRONGLY CONNECTED COMPONENT
5300C            WHOSE COMPONENT NUMBER IS (CN(.
5310C
5320      IMPLICIT INTEGER (A-Z)
5330      REAL CYCLEM
5340      COMMON /ALIST/ N,M,AI(31), A(480)
5350      COMMON /CLIST/ CI(30), CA(900)
5360      COMMON /CYCLE/ S,CY,T1,T2,CE,EDGE,SELECT
5370      COMMON /SCCMP/ VKC,VKMAX,CN,NCN,VK(30), SCC(30)
5380      COMMON /CRRNT/ BI(30), BP(30), STATE(30)
5390      COMMON /WORKS/ CS(30), BS(30), CP(30), FLAG(30), B(900),  CAB(900)
5400      COMMON /STOCK/ CYCLEM(430,450),INDEXM(30,30),  LENG(430),IPRINT
5410C
5420      MAS=1
5430C
5440      C=0
5450      V=S
5460      CYY=0
```

```
5470C
5480C      **  STACKING THE MOST RECENTLY REACHED VERTEX V  **
5490C
5500   100 CONTINUE
5510       C=C+1
5520       CS(C)=V
5530       FLAG(V)=0
5540       STATE(V)=1
5550       CP(V)=CI(V)
5560C
5570C      **  EXPLORING VERTEX W ADJACENT TO VERTEX V  **
5580C
5590   200 CONTINUE
5600       IF( MAS.EQ.0 ) GOTO 1000
5610       IF(   C.EQ.1 ) GOTO 201
5620       IF( CYY.EQ.1 ) GOTO 300
5630   201 CYY=0
5640  1000 T=CP(V)
5650       W=CA(T)
5660       IF( W.EQ.0 ) GOTO 300
5670       CP(V)=T+1
5680       IF( W.EQ.S ) GOTO 250
5690       IF( STATE(W).NE.0 ) GOTO 200
5700       V=W
5710       GOTO 100
5720C
5730C      **  A NEW CYCLE  IS  FOUND  **
5740C
5750   250 CONTINUE
5760       CY=CY+1
5770       CYY=1
5780       CS(C+1)=CS(1)
5790       LENG(CE+CY)=C
5800       IF( IPRINT.NE.1 ) GOTO 12
5810       WRITE(6,10) (CS(I),I=1,C+1)
5820    10 FORMAT(5X,25I4)
5830    12 CONTINUE
5840       DO 13 J=1,EDGE
5850    13 CYCLEM( CE+CY, J )=0
5860       DO 14 I=1,C
5870       CYCLEM( CE+CY , INDEXM(CS(I),CS(I+1)) )=1
5880    14 CONTINUE
5890       IF( CY.EQ.SELECT ) GOTO 900
5900       FLAG(V)=1
5910       GOTO 200
5920C
5930   900 S=N
5940       RETURN
5950C
5960C
5970C      **  BACKTRACKING  **
5980C
5990   300 CONTINUE
6000       C=C-1
6010       IF( C.EQ.0 ) RETURN
6020C
6030       W=CS(C)
6040       IF( FLAG(V).NE.0 ) GOTO 400
6050C
6060C      **  CONSTRUCTING  LIST B(.)  **
6070C
6080       T=CI(V)-1
6090   350 CONTINUE
6100       T=T+1
6110       X=CA(T)
6120       IF( X.EQ.0 ) GOTO 500
6130       IF( CAB(T).NE.0 ) GOTO 350
6140       TT=BP(X)
6150       B(TT)=T
6160       CAB(T)=V
6170       BP(X)=TT+1
6180       GOTO 350
```

```
6190C
6200C      **  UNBLOCKING THE BLOCKED( STATE(.)=1 ) VERTICES  **
6210C
6220   400 CONTINUE
6230       D=1
6240   420 CONTINUE
6250       BS(D)=V
6260       STATE(V)=0
6270   450 CONTINUE
6280       TT=BP(V)
6290       IF( TT.EQ.BI(V) ) GOTO 470
6300       TT=TT-1
6310       BP(V)=TT
6320       T=B(TT)
6330       X=CAB(T)
6340       CAB(T)=0
6350       IF( STATE(X).EQ.0 ) GOTO 450
6360       D=D+1
6370       V=X
6380       GOTO 420
6390   470 CONTINUE
6400       D=D-1
6410       IF( D.EQ.0 ) GOTO 490
6420       V=BS(D)
6430       GOTO 450
6440C
6450   490 CONTINUE
6460       FLAG(W)=1
6470   500 CONTINUE
6480       V=W
6490       GOTO 200
6500C
6510       END
```

【サブルーチン L　P】

```
6520C
6530C
6540C*************************************************************************
6550       SUBROUTINE LP(M,N,A,B,C,IRANGE,   NNS,BBS,ZBAR)
6560C
6570       DIMENSION A(430,450),B(430),C(450),BS(430),BSO(430),BBS(430)
6580       DIMENSION NS(450),NNS(450),CS(450),ISEEK(100)
6590       COMMON /TIMEB/ ITB1,ITB2
6600C
6610       CALL FPARAM(1,132)
6620       CALL LONGIO(6,132)
6630C
6640       CALL CPTIME(ISTART)
6650C
6660C      **      INITIALIZATION      **
6670C
6680       DO 30 I=1,N
6690       CS(I)=0.0
6700       NNS(I)=0
6710    30 CONTINUE
6720       DO 40 I=1,M
6730       BS(I)=0.0
6740       BBS(I)=0.0
6750       BSO(I)=B(I)
6760    40 CONTINUE
6770       ICOUNT=0
6780C
6790C      WRITE (6,830)(I,I=1,N)
6800C 830 FORMAT(//5X,"ORIGINAL  PROBLEM"//
6810C     &           5X,"OBJECTIVE FUNCTION",/( 30X,10(3X,"X(",I3,") " ) ) )
6820C      WRITE(6,840)(C(I),I=1,N)
6830C 840 FORMAT(17X,"F =",10X,10(2X,F8.1)/(30X,10(2X,F8.1)/) )
6840C      WRITE(6,850)
6850C 850 FORMAT(/5X,"CONSTRAINT EQUATIONS"/)
```

```
6860C      DO 70 I=1,M
6870C      WRITE (6,860) I,B(I),(A(I,J),J=1,N)
6880C  860 FORMAT (8X,"O (= G(",I3,")=",11(2X,F8.1)/(30X,11(2X,F8.1)/)')
6890C   70 CONTINUE
6900C
6910C      **      FOR  C )= 0      **
6920C
6930       DO 90 J=1,N
6940       CS(J)=C(J)
6950       IF (C(J).GT.0.0) GO TO 90
6960       C(J)=-C(J)
6970       DO 80 I=1,M
6980       B(I)=B(I)+A(I,J)
6990    80 A(I,J)=-A(I,J)
7000    90 CONTINUE
7010C
7020       IB=0
7030C
7040C      **      ZBAR - SETTING      **
7050C
7060       ZBAR=0.0
7070       DO 100 J=1,N
7080       IF( CS(J).GT.0.0 ) GOTO 100
7090       ZBAR=ZBAR+C(J)
7100   100 CONTINUE
7110C
7120C         WRITE (6,870)(I,I=1,N)
7130C     870 FORMAT(//5X,"MODIFIED  PROBLEM    FOR  THIS ALGORITHM"//
7140C       &           5X,"OBJECTIVE FUNCTION",/( 30X,10(3X,"X(",I3,") " ) ) )
7150C         WRITE(6,840)(C(I) ,I=1,N)
7160C         WRITE(6,850)
7170C         DO 140 I=1,M
7180C         WRITE (6,860) I,B(I),(A(I,J),J=1,N)
7190C     140 CONTINUE
7200C
7210C      **      SEARCHING  INITIAL VARIABLE TO BE FIXED      **
7220C
7230       IFL=-1
7240       L=N
7250C
7260   460 DO 400 I=1,N
7270       IF( CS(L).LE.0.0 ) GOTO 440
7280       L=L-1
7290       IF( L.EQ.0 ) GOTO 960
7300   400 CONTINUE
7310   440 DO 450 J=1,M
7320       B1=BSO(J)-A(J,L)
7330       IF( B1.GE.0.0 ) GOTO 450
7340       L=L-1
7350       IF( L.EQ.0 ) GOTO 960
7360       GOTO 460
7370   450 CONTINUE
7380C      WRITE(6,930) L
7390C  930 FORMAT(/5X,"INITIAL  L =",I5/)
7400       GOTO 410
7410C
7420   960 WRITE(6,970)
7430   970 FORMAT(/5X,"FEASIBLE SOLUTION  IS NOT  FOUND" )
7440       GOTO 280
7450C
7460C      **      SEARCHING  OTHER  VARIABLES TO BE FIXED      **
7470C
7480   150 IF( IFL.NE.-2 ) GOTO 430
7490       IFL=0
7500       DO 420 I=1,N
7510       IF( NNS(I).NE.0 ) GOTO 420
7520       IFL=IFL+1
7530       ISEEK(IFL)=I
7540   420 CONTINUE
7550       IFLO=IFL
7560C
7570       CALL CPTIME(ITO)
7580       IA2=ITO-ISTART
```

```
7590C
7600C      **      CALCULATING      **
7610C
7620   430 CALL CPTIME(ITO)
7630       L=ISEEK(IFL)
7640       IF( CS(L).GT.0.0 ) GOTO 270
7650       DO 470 J=1,M
7660       B1=BSO(J)-A(J,L)
7670       IF( B1.LT.0.0 ) GOTO 270
7680   470 CONTINUE
7690   410 DO 160 I=1,N
7700   160 NS(I)=0
7710       DO 170 J=1,M
7720   170 BS(J)=BSO(J)
7730       ZS=0.0
7740       DO 180 J=1,M
7750   180 BS(J)=BS(J)-A(J,L)
7760       NS(L)=L
7770       DO 220 I=1,N
7780       IN=N+1-I
7790       IF( IN.EQ.L ) GOTO 220
7800       IF( CS(IN).GT.0.0 ) GOTO 220
7810       DO 190 J=1,M
7820       J1=J
7830       BS(J)=BS(J)-A(J,IN)
7840       IF( BS(J).LT.0.0 ) GOTO 200
7850   190 CONTINUE
7860       NS(IN)=IN
7870       GOTO 220
7880   200 DO 210 JJ=1,J1
7890   210 BS(JJ)=BS(JJ)+A(JJ,IN)
7900       ZS=ZS+C(IN)
7910       IF( ZS.GT.ZBAR ) GOTO 270
7920   220 CONTINUE
7930       DO 240 I=1,N
7940       IF( NS(I).NE.NNS(I) ) GOTO 250
7950   240 CONTINUE
7960       GOTO 270
7970   250 ZBAR=ZS
7980       DO 260 I=1,N
7990   260 NNS(I)=NS(I)
8000       DO 290 I=1,M
8010   290 BBS(I)=BS(I)
8020C            WRITE(6,911) ZBAR,IFL
8030C       911 FORMAT(/5X,"ZS    =",F14.8 ,I15/)
8040C            WRITE(6,920) (NS(I),I=1,N)
8050C       920 FORMAT(2X,40I3)
8060   270 IFL=IFL-1
8070       ICOUNT=ICOUNT+1
8080       IF( IFL.EQ.-2 ) GOTO 150
8090       CALL CPTIME(IT1)
8100       IB=IB+IT1-ITO
8110C      WRITE(6,998) IT1-ITO
8120C  998 FORMAT( 5X,"SOLVING  L.P.          CPU =",I15,"    M SEC")
8130       IF( IFL.EQ.IFLO-IRANGE ) GOTO 280
8140       IF( IFL.EQ.0 ) GOTO 280
8150       GOTO 150
8160C
8170   280 DO 300 J=1,N
8180       IF (CS(J).GT.0.0) GO TO 300
8190       C(J)=-C(J)
8200       DO 310 I=1,M
8210       A(I,J)=-A(I,J)
8220   310 B(I)=B(I)-A(I,J)

8220   310 B(I)=B(I)-A(I,J)
8230   300 CONTINUE
8240C
8250       IB=IB/(ICOUNT-1)
8260       CALL CPTIME(ITB2)
8270       ITB2=ITB2-ISTART
8280       ITB1=IA2+IB*IFLO
```

```
8290C
8300C      WRITE(6,910) ZBAR
8310C 910 FORMAT(/5X,"ZBAR =",F13.3/)
8320C      WRITE(6,920) (NNS(I),I=1,N)
8330C
8340       RETURN
8350       END
```